Spezial- Magazin für den Home-Computer-Besitzer

computronic

Doppelausgabe
Nov./Dezember 84 12/1. Jahrgang

DM 6,50
öS 55
s.Fr 6,50

Sonderteil Commodore 64 Topprogramme

Jetzt: 88 Seiten Software aktuell

Commodore 64
VC-20
Atari
TI-99
ZX-Spectrum
ZX-81
Apple II

Kompletter Forth-Kurs

Viele Überraschungen!!

Zum Programmieren

11 SUPER-SPIELE

+ Anwenderprogramme

Möchten Sie als Software-Autor für den TRONIC-Verlag tätig werden?

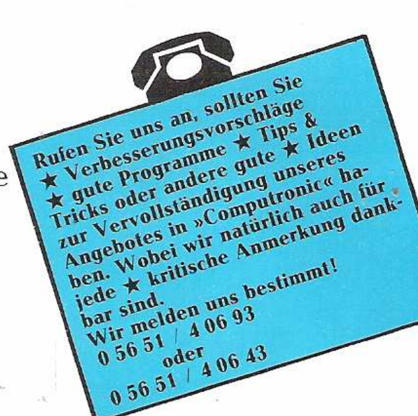
Wir würden uns freuen, in Ihnen einen Ansprechpartner für eine künftige intensive Zusammenarbeit zu finden. Der TRONIC-Verlag vereinbart mit seinen künftigen Software-Autoren pro veröffentlichte volle Seite (einschl. Programmbeschreibung) **ein Honorar von DM 120,–.**

Dieses Entgelt wird fällig, wenn die Redaktion des Verlages sich für eine Veröffentlichung entscheidet. Die Auszahlung erfolgt also nicht erst nach Veröffentlichung in einer unserer Ausgaben, sondern früher.

Der Verlag wird vom Autor berechtigt, seine Manuskripte (Programme) zur Darstellung im Heft heranzuziehen und abzudrucken.

Einzusenden sind:

- Programmbeschreibung
- bespielte Cassette oder Diskette
- Listing (mit Copyright)
- Freiumschlag



Der Autor erklärt sich mit der Lieferung seines Programmes oder seiner Beiträge ausdrücklich bereit, die Verwertung durch den Verlag freizugeben, d. h. er überträgt nicht nur die Nutzung, sondern auch die Übereignung des Computerprogrammes und der Beiträge.

Der Autor verpflichtet sich nur solche Programme anzubieten, die eigene Entwicklungen des Autors sind.

Mit der Veröffentlichung oder dem Anlauf des Programmes und der Beiträge ist es dem Verlag gestattet, auch eine anderweitige bzw. weitergehende Verwertung vorzunehmen, da der Autor dem Verlag das Copyright gegen Honorar gestattet hat. Die Verwertung durch den Verlag ist unbeschränkt und unwiderruflich, wenn nicht 10 Tage nach Zusendung der Unterlagen durch den Autor widersprochen wird.

Haben Sie Interesse? Haben Sie noch Fragen?

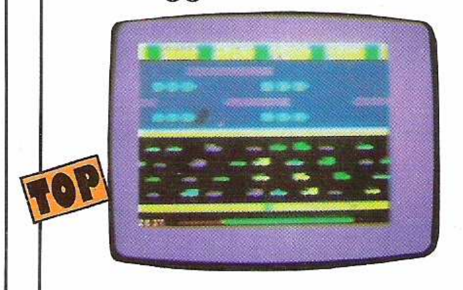
Setzen Sie sich telefonisch mit unserer Redaktion in Verbindung!

TRONIC-VERLAG

DIE REDAKTION

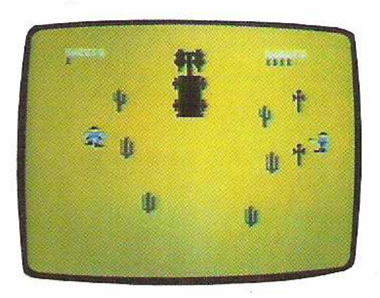
Aus dem Inhalt:

Frogger



ZX-Spectrum 48 K

High-Noon



Commodore 64

Skeet



Commodore 64

Cave-Man



TI-99 mit und ohne Ext.-Basic

In diesem Heft

11/12 1984

Software

Aktuelles

Forth-Kurs

Auf vielfachen Wunsch hier unser kompletter Kurs

Vorstellung der Programmiersprache

Wer schon in Basic programmiert hat wird wissen, daß es hin und wieder Anwendungen gibt, die recht schwierig zu programmieren sind. Für solche Fälle hat man bei großen Rechnern die Möglichkeit, auf andere Sprachen zurückzugreifen. Es gibt *FORTRAN* für mathematische Problemlösungen, *COBOL* für kommerzielle Zwecke, *ASSEMBLER* für zeitkritische Aufgaben, *BASIC* für die allgemeinen Problemlösungen auf Microcomputern.

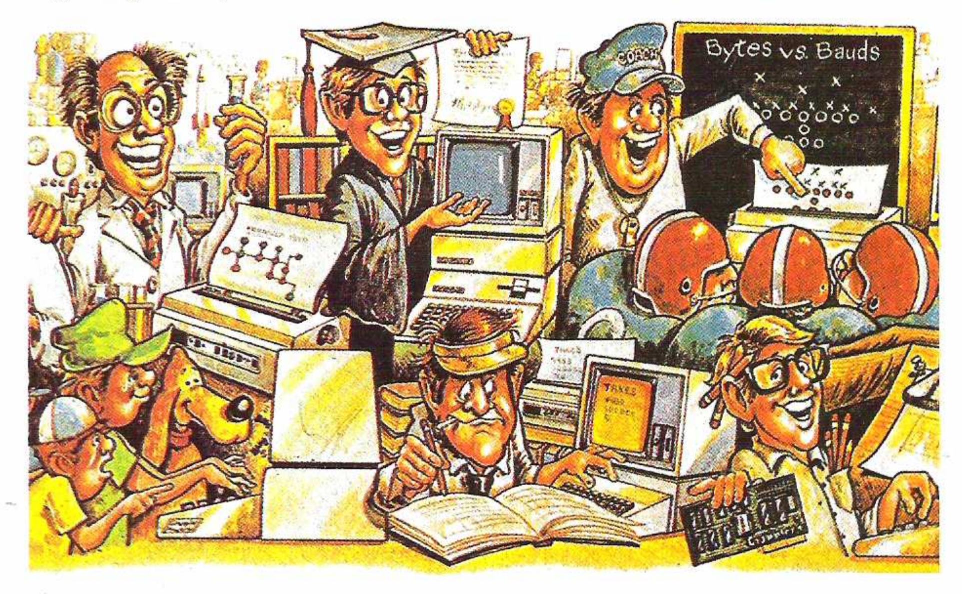
FORTH gehört zu der jüngsten Generation von Programmiersprachen. Versucht wurde, alle Vorteile der renommierten Programmiersprachen in *FORTH* zu implementieren, ohne aber deren „Fehler" mit zu übernehmen.

FORTH hat schon in seiner Struktur einige markante Vorteile, die besonders bei den Microcomputern von Vorteil sind:

- *FORTH* ist auf fast allen Microcomputern implementierbar und benötigt nur wenig Speicherplatz.
- *FORTH*-Programme sind wesentlich kürzer als *BASIC*- und *ASSEMBLER*-Programme.
- *FORTH* ist in der Regel 10mal schneller als *BASIC*.
- *FORTH* kann auch mit Kassettenrecorder betrieben werden. Vorteilhaft ist aber ein Floppy-Laufwerk.
- *FORTH* ist problemlos erweiterbar.
- *FORTH*-„Hochsprache" und *FORTH*-„Assembler" lassen sich beliebig mischen.
- *FORTH* erlaubt eine strukturierte Programmierung.

FORTH ist ursprünglich für die Programmierung von Steuerungsaufgaben entwickelt worden. Da die Sprache leicht erweiterbar ist, kann sie für fast alle Anwendungen verwandt werden. Durch das Implementieren einiger Grafik-Befehle wird aus der Prozeßsprache schnell eine geeignete Sprache für Spiele und Grafik-Anwendungen. Jeder kann *FORTH* um die Befehle erweitern, die er für seine Anwendungen gerade benötigt. Trotz zahlreicher Erweiterungen sind *FORTH*-Programme sehr leicht auf einen anderen Rechner zu übertragen. Fehlende Worte (Befehle) können einfach implementiert werden.

Wir hoffen, daß es uns gelungen ist, einige Vorteile von *FORTH* darzustellen. Auf den folgenden Seiten werden Sie schrittweise mit den Eigenheiten der Sprache vertraut gemacht. Es ist schade, daß bisher kaum eine Zeitschrift näher auf *FORTH* eingegangen ist. Obwohl es sicher eine der leistungsfähigsten Sprachen für Microcomputer ist und immer mehr Freunde findet.



Einleitung

FORTH ist ein selbständiges Sprachsystem, das sowohl Compiler als auch Interpreter ist und sein eigenes Betriebssystem beinhaltet. Das Besondere an *FORTH* ist, daß mit dem „Stapel" gearbeitet wird. Wer schon einmal in Assembler programmiert hat, kennt die Arbeitsweise des Stapels. Der Stapel arbeitet nach dem *LIFO-System*. Dies bedeutet, etwas wird abgelegt und kann später wieder heruntergenommen werden. *FORTH* besitzt zwei Stapel. Den Daten- und den Return-Stapel. Auf die Funktion wird später noch eingegangen.

Die Verwendung eines Stapels wäre eigentlich nichts Neues. Jeder HP-Taschenrechner verwendet einen Stapel zum Abspeichern von Zahlen. Der Unterschied zu den HP-Rechnern ist hier das Wörterbuch, welches auch als Stapel ausgelegt ist.

Die Sprache besteht aus Worten, die nach ihrer Definition auf dem Definitionsstapel, dem Wörterbuch, abgelegt werden. Bei der Erstellung eines Wortes können schon vorherige Definitionen verwendet werden. Ein *Forth-Programm* besteht also in der Regel aus zahlreichen Definitionen, die sich gegenseitig aufrufen.

Um Wörter wieder löschen zu können, benutzt man den Befehl >FORGET<. Der Aufruf >FORGET TEST< löscht alle Worte, beginnend bei >TEST< bis zum obersten Eintrag in den Stapel. Das Löschen eines einzigen Wortes im Stapel ist nicht möglich und wäre auch nicht sinnvoll. Denn spätere Definitionen könnten nicht mehr benutzt werden. Es gibt aber auch die Möglichkeit, ein Wort mehrmals zu definieren. In diesem Fall wird immer das zuletzt definierte Wort ausgeführt.

Berechnungen erfolgen in *FORTH* über den Datenstapel, den wir auch als STACK bezeichnen können. Da alle Operanden über den Stack geleitet werden, ergibt sich die etwas ungewohnte Rechenschreibweise >UPN< (Umgekehrte Polnische Notation). Auch die schon genannten HP-Taschenrechner verwenden diese Art der Eingabe. Um beispielsweise die Zahlen 5 und 7 zu addieren, muß folgendes eingegeben werden:

in FORTH: 5 7 + *in BASIC:* 5 + 7

Nach der Eingabe dieses Beispiels werden die beiden Summanden vom Stack geholt und addiert. Das Ergebnis wird wieder auf den Stack geschrieben. Man kann dies leicht überprüfen, indem man sich das Ergebnis durch den Befehl >"."< ausgeben läßt. Der Befehl >"."< holt die oberste Zahl

weiter Seite 31



Der PaperTiger von Dataproducts ist gewachsen!

Seit seiner Vorstellung im April 1984 ist das Modell 8010/11 aus der Paper-Tiger-Serie von Dataproducts, dem weltgrößten, unabhängigem Druckerhersteller, Maßstab für professionelle und modernste Matrixdrucker im mittleren Preisbereich.

Jetzt ist mit dem Modell 8020/21 die breitere Version verfügbar *(132 Zeichen/Schreibbreite)*, die vom Design und den Eigenschaften mit dem Modell 8010/11 identisch ist.

Als sehr wichtig für Anwender hat sich bei der Serie die Ausstattung der Drucker mit einem gemeinsamen parallelen und seriellen Interface erwiesen, desgleichen, daß die Modelle 8011 und 8021 speziell für den IBM-PC und die Kompatiblen entwickelt wurden.

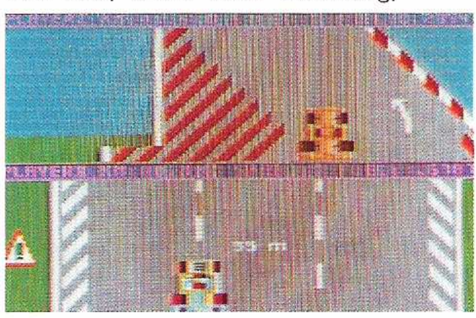
Dialogverkehr statt „Mäuseklavier" – ist wohl die bemerkenswerteste Eigenschaft der Drucker. Denn auf sämtliche Funktionseinheiten kann über die Sensortasten zugegriffen werden. Damit entfällt das manchmal nervenaufreibende Manipulieren an den üblicherweise eingebauten DIP-Schaltern. Der Anwender kann sich per Tastendruck eine Übersicht über die aktuellen Betriebsparameter ausdrucken lassen. Die angezeigten Parameter (insgesamt 22) lassen sich dann über einen einfach zu erlernenden Dialog beliebig modifizieren. Voreinstellungen bestimmter Funktionen lassen sich speichern, so daß sie nach dem Ausschalten des Druckers auch erhalten bleiben. Dafür sorgt ein eingebauter batteriegepufferter RAM-Speicher.

Dataproducts GmbH, Frankfurter Str 172–176, 6078 Neu Isenburg,

Atari „HIGHWAY DUELL"

Mit dem Spiel-Programm „Highway Duell" für den Atari 400/800 und die XL-Modelle präsentiert der große Hamburger Homecomputer-Software- und Zubehör-Anbieter eine rein deutsche Software-Entwicklung.

Die beiden Münsteraner Abiturienten Julian Reschke und Andreas Wiethoff programmierten dieses Autorennen in Maschinensprache unter Anleitung von Experten. Vorabmuster wurden von Kindern und Jugendlichen auf Schwierigkeiten und Spielwitz geprüft. Bis zur Fertigstellung des Programms und der Produktion mit einem kaum zu „knackenden" Kopierschutz verging fast ein halbes Jahr.



In einem schwierigen Straßenrennen mit ausgezeichneter Graphik kämpfen Sie gegen Ihren Mitspieler oder den Computer um den Sieg. Baustellen müssen langsam passiert werden, Hindernisfahrzeugen ausgewichen werden. Kleine, benzinfressende Wesen zwingen Sie zum Tankaufenthalt. Wer wird Siegen? *Gesehen bei: Dynamics marketing GmbH, 2000 Hamburg 1.*

Unseriöse Machenschaften

Wer mit Computern zu tun hat, hat sich meist auch mit *Softwarepiraten* herumzuschlagen. Als Softwarepiraten bezeichnen wir solche Leute, die ohne Genehmigung Programme kopieren und weiterverkaufen. Die Inhaber der Firma *R & S Computerorganisation* machten sich dieses zunutze. Sie verschickten an sogenannte Piraten Abmahnungen. Die Adressen holten sie sich aus Anzeigen in Fachzeitschriften. Der Inhalt dieser Briefe lautete – „Tests haben ergeben, daß Sie mit Computerprogrammen handeln, deren Urheberrecht Sie nicht besitzen. Dabei wurde festgestellt, daß es sich um Raubkopien handelt. Mit dem vorstehenden Sachverhalt verstoßen Sie gegen die Bestimmungen des § 1 UWG. Der von uns geschätzte Schaden beläuft sich auf ca. 60 000,- DM (sechzigtausend)!! Zur Vermeidung gerichtlicher Schritte fordern wir Sie auf, binnen 5 Tagen zu erklären, daß Sie in Zukunft keine Computerprogramme über die Sie kein Urheberrecht besitzen, zu vertreiben. Zur Begleichung unserer Aufwendungen erwarten wir, daß Sie den Betrag von 300,- DM laut Kostenrechnung *in bar* der Unterlassungserklärung beilegen." – Dann folgt eine Anmerkung, die jeden stutzig machen sollte. – „Einschreiben werden nicht in Postfächer zugestellt. Bitte verwenden Sie nur einen Standardbrief." – Dieses ist nicht richtig! Wer einen Brief an diese Adresse geschickt und Geld beigefügt hat, wird nun in die Röhre gucken. Was sich herausstellte war folgendes: Die R & S Computerorganisation ist eine Scheinfirma die kein Postfach sondern nur eine Postlagerkarte hatte. Diese Firma ist auch nicht auf dem Gewerbeamt oder als Abmahnverein in Berlin bekannt. Eigene Nachforschungen haben ergeben, daß die Gewerbepolizei unter der Leitung von Hauptkommissar Müller bereits auf diese Firma aufmerksam gemacht worden ist.
Unsere Rückfrage bei der Post hatte ergeben, daß dort bereits eine große Anzahl von Briefen eingegangen war und dort lagerte. Die Polizei, die sich nach einem Hinweis bemühte die Täter der Postlagerkarte zu fassen, konnte jedoch nur einen Jugendlichen festnehmen. Bei ihm wurden insgesamt 106 Briefe!! beschlagnahmt. Diese wurden von der Oberstaatsanwaltschaft Berlin ungeöffnet an ihre Absender zurückgesandt. Sechs Briefe, deren Absender nicht zu ermitteln waren, wurden geöffnet. Nach Auskunft von Oberstaatsanwalt Dr. Weimann hatte sich kein Geld in den Briefen befunden. Alle Briefe wurden ebenfalls an die Absender zurückgesandt.
Die Reaktion auf solche unseriöse Machenschaften zeigt aber deutlich, wie verunsichert die Empfänger eigentlich waren. Eines sollte wohl allen klar sein, wer Programme vertreibt, deren Urheberrechte jedoch bei anderen Personen oder Gesellschaften liegen, macht sich strafbar.

Ein Kleiner, der doch schon so groß ist!!!

Unter dem Namen GTS 80 stellt Grand Tree Systems Computer, Hardware und Software Gesellschaft, eine neue Reihe von netzwerkfähigen CP/M-Rechnern vor. Die Rechner zeichnen sich durch alle nur denkbaren Variationen bezüglich der Massenspeicher sowie die enorme Verarbeitungsgeschwindigkeit aus. Der Rechner verarbeitet ohne Probleme Software und Datenformate, wie sie auf dem „Wave Mate Bullet"* gefahren werden. An Schnittstellen werden standardmäßig angeboten:
2 x RS 232 C
2 x RS 422
1 x Centronix parallel
Sasi Bus (Winchester-Anschluß)
ECB-Bus
4 x 5¼-" Floppy-Anschluß
4 x 8-" Floppy-Anschluß
Alle Schnittstellen sind bereits anschlußfertig mit Steckern herausgeführt.
Den GTS 80 gibt es ebenso in einer Backup Version mit zwei Festplatten, wobei die zweite Festplatte nur zur Datensicherung eingesetzt wird.
Die Kapazitäten der Festplatten:
10 MB formatiert
43 MB formatiert
Die Krönung aller GTS 80-Rechner ist der GTS 80m, m für Master und das besagt wiederum, daß unter Verwendung eines GTSm und bis zu 32 GTSn das wohl im Moment leistungsfähigste Netzwerk installiert werden kann. Standard Terminals werden durch nachrüstbare Zusatzplatinen (Platinen für das GTS-Netzwerk) zu vollwertigen Arbeitsplätzen innerhalb des Netzverbundes (Slave Workstation).
Als Betriebssystem wird hierfür BRIDOS verwendet.
Bridos, ein Betriebssystem mit UNIX-Feautures, emuliert CP/M, ohne jedoch wie unter CP/M die Directory-Einträge zu limitieren.
Das Netzwerk kann jederzeit nach Terminvereinbarung besichtigt und getestet werden.
Der GTS 80 mit 2 x 800 KB Floppy Drive einschließlich CP/M 3-Lizenz, 128 KRAM, DMA und 2 x SIO kostet DM 4736,– zuzügl. MwSt. Ein Terminal mit 14"-Bildschirm in Bernstein mit 10 Emulationen kostet 1398,– DM zuzügl. MwSt.

* Wave Mate Bullet is the registered Trade Mark von Wave Mate Bullet.

Grand Tree Systems Computer
Hardware und Software Gesellschaft
6472 Altenstadt
Tel. 0 60 47 / 24 99 Tex. 4 184 951

THORN EMI PLC

größter britischer Elektronik-Konzern, nutzt seine langjährigen Erfahrungen auf dem Gebiet der Informationstechnik und hat jetzt weltweit den neuen Geschäftsbereich „Computer Software" gegründet.
In Deutschland heißt das neue Unternehmen *THORN EMI Computer Software GmbH.* Als Anbieter von deutschsprachiger Standardsoftware für alle führenden Personal- und Home-Computer befindet sich THORN EMI seit Anfang Juni 1984 auf dem Markt.
Das Angebot wird hauptsächlich Business Software, Erziehungsprogramme und anspruchsvolle Unterhaltungsspiele für Jugendliche und Erwachsene umfassen. Die Distribution erfolgt im gesamten Bundesgebiet über ein Netz von Handelsvertretern. THORN EMI besitzt sowohl eigene Softwarehäuser als auch Rechte an fremder Qualitätssoftware. Händlern und Kunden wird gleichermaßen ein kompletter Beratungsservice in allen Bereichen der Computersoftware geboten.
Geschäftsführer Klaus D. Geiser (51), vorher lange Jahre in leitender Position bei Philips und 3M, sieht große Chancen auf dem stark expandierenden Markt der Standard-Software. Der Jahresumsatz des Geschäftsbereichs Computersoftware der *THORN EMI* liegt weltweit bei 5 Mio £. Innerhalb der nächsten drei Jahre soll diese Summe verzehnfacht werden.

Im Test: Micro Power 2000

Schnelles Doppellaufwerk für den Commodore

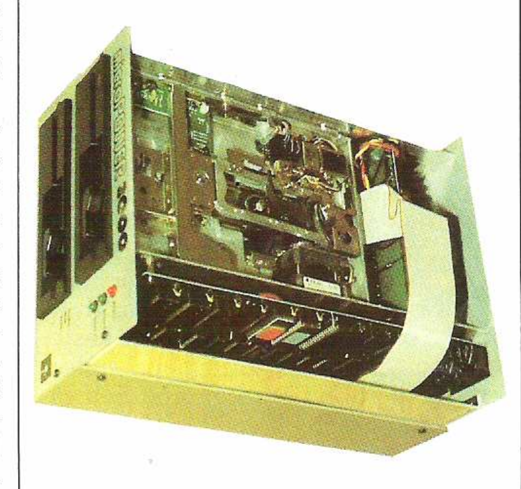
Als ein kompaktes Doppellaufwerk in einem Alugehäuse fanden wir die MICRO POWER 2000 nach dem Auspakken vor. Das erste was uns auffiel, war das ansprechende Handbuch, in dem alles aufgeführt ist, was ein Benutzer der MICRO POWER 2000 wissen muß.

Die MICRO POWER 2000 ist eine intelligente Doppeldiskettenstation, die dem Rechner keinen Speicherplatz abverlangt. Auf der Frontplatte befinden sich zwei Einschübe für 5 124-Zoll-Disketten. Eine rote Leuchtdiode ist für die Netzkontrolle, zwei grüne Leuchtdioden sind als Laufwerkkontrolle gedacht. Auf der Rückseite befinden sich die Anschlüsse für zwei serielle Buchsen und den IEEE 488-Bus. Neben dem Netzanschluß und der Sicherung befindet sich der Ein- und Ausschalter. Die beiden seriellen Anschlüsse sind zum Anschluß an den VC-20, CBM 64, SX 64 und den Commodore Plus 4. Des weiteren können alle Drucker über den seriellen Port angeschlossen werden. Alle Commodore Rechner im Homecomputer-Bereich und Rechner der Serien 2000, 3000, 4000, 8000, 600 und 700 können die MICRO POWER 2000 über den IEEE-Bus ansteuern.

Somit hat die MICRO POWER 2000 allen Laufwerken im Homecomputer-Bereich eines voraus: die Umsteiger auf den PC-Bereich müssen nicht erst ein neues Laufwerk kaufen.

Alle Befehle, die es für die Floppy 1541 gibt, können für das Doppellaufwerk benutzt werden. Auch einfache Befehle wie: – Open 1,8,15,"D1=0" – erlauben ein einfaches Duplizieren von ganzen Disketten. Das Einladen von Programmen ist genauso einfach wie bei der 1541 – Load"0:Name",8 oder Load"1:Name",8 –. Ebenso einfach lassen sich einzelne Dateien kopieren – Open 1,8,15,"c1:name=0:Name" –, dieser Befehl bewirkt, daß eine Einzeldatei von Laufwerk 0 auf Laufwerk 1 kopiert wird. Durch Tauschen der Adressen – C0=1 – wird von 1 nach 0 kopiert.

Daß Geschwindigkeit keine Hexerei ist, beweist die Kopierzeit einer 664-Block-Diskette von nur 1,40 Minuten. Aber noch schneller arbeitet man mit dem IEEE 488-Bus. Das Einladen von Programmen ist ohne IEEE-Bus fast genauso langsam wie bei der 1541. Mit dem IEEE-Bus geht dieses aber 6mal schneller als bei der 1541. Auf der beigelegten Demodiskette ist Basic 4.0 abgespeichert. Dieses kann jederzeit eingeladen werden, so daß dem Benutzer viele Wege (Befehle) offenstehen.



Natürlich gibt es einige wenige Programme, die auf der MICRO POWER 2000 nicht auf Anhieb laufen. Solche Programme sind aber in der Minderzahl, so daß dem fachkundigen Benutzer keine Probleme entstehen, sie umzuschreiben. Die von uns eingesetzten Programme Textomat, Datamat, Multidata, Multiplan und Spielprogramme haben keine Probleme ergeben.

Zusammenfassung

Die MICRO POWER 2000 ist ein Doppellaufwerk, das dem Benutzer keine Wünsche offen läßt. Die Vorteile für das Doppellaufwerk liegen klar auf der Hand. Wenn der Benutzer aus dem Homecomputer-Bereich in den PC-Bereich umsteigen will, braucht kein neues Laufwerk erworben zu werden. Die MICRO POWER 2000 ist auch im PC-Bereich einsetzbar. Dauertests haben keine Beanstandungen ergeben. Auch das Warmwerden, wie bei der 1541, ist nicht gegeben. Die Verarbeitung sammelt weitere Pluspunkte. Die MICRO POWER 2000 bringt viel Floppykomfort, die alle anderen Floppystationen für den Homecomputer-Bereich in den Schatten stellt. Die MICRO POWER 2000 ist im Fachhandel oder bei RMC Systems in Oberhausen zum Preis von 2695,00 DM erhältlich.

Rolf Freitag

Internationale Commodore Fachausstellung

Alles Rund um den Commodore

Anfang September war in Frankfurt die 4. Commodore Fachausstellung. Auf 4000 qm boten 63 Aussteller aus Schweden, der Schweiz, England, Österreich und Deutschland Software, Hardware, Erweiterungen und Literatur rund um den Commodore an. Auch Seminare, Referate und Workshops wurden auf der Commodore Fachausstellung '84 abgehalten. So konnte jeder Besucher seinen Informationswünschen nachgehen. Commodore hat sich einige Mühe gegeben, um wirkungsvolle *„Software-Munition"* auffahren zu können.

Mindestens ebenso reizvoll wie die ei-



Commodore SX 64

gene Leistungsschau des Veranstalters, war das Leistungsangebot der 63 Aussteller. Die Angebote reichten von der Meßwerterfassung über die Fertigungssteuerung, bis hin zum Radsporttraining. Eine Reihe von Peripherie-Firmen hatten ebenfalls attraktive Neuerungen auf der Messe. So etwa eine Platine, mit deren Hilfe alle Programme für die Systeme *CBM 4032, 8032* und *8096* auf Rechnern der Modellreihe 700 gefahren werden können. Die Montage ist in wenigen Minuten möglich und – es geht kein Speicherplatz verloren.

Ein Mekka für Computerfreunde
Zu einem Mekka der Computerfreunde hat sich die Commodoreshow in den letzten 3 Ausstellungen aufgetan. So haben sich in diesem Jahr mehr Verlage auf der Messe ein Stelldichein gegeben. Unter anderem so bekannte Namen wie Langenscheidt, Otto Maier und Georg Westermann. Diese Verlage präsentierten Lernprogramme und Anleitungsbücher für Home- und Profi-User.
Das Arbeitsamt und die Volkshochschule waren mit einem Stand vertreten, wo sich ratsuchende Teens und Twens, Schüler und Studenten Informationen einholten.
Großes Gedränge herrschte an den Ständen bei: *Software Express, RMC Systems, DATA Becker* und natürlich *Commodore.* Bei *RMC Systems* aus

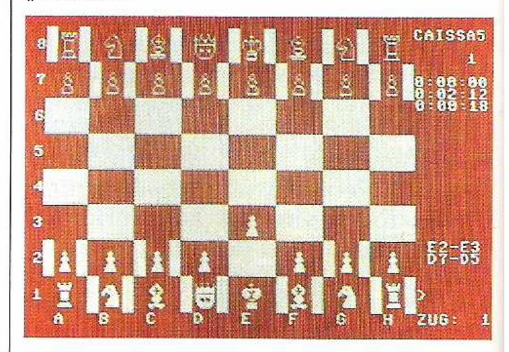


Oberhausen ist es uns gelungen, die schon lange angekündigte Micro Power 2000 zu bekommen und zu testen. Dort sahen wir auch einen der ersten Commodore SX 64 mit Doppellaufwerk. Jim Butterfield, auch „Commodor Papst" genannt, bekam beim Anblick des SX 64 mit Doppellaufwerk einen interessanten Gesichtsausdruck, zumal Commodore bis heute nicht einen tragbaren Computer mit Doppellaufwerk herausgegeben hat. Eine Anmerkung sei allerdings erlaubt, der SX 64 hat kein *1541-Laufwerk,* sondern ein *kompatibles Laufwerk.*
Weltweite Verbindungen wurden per Akustikkoppler auf dem Stand von *Software Express* aus Düsseldorf hergestellt. Die Besitzer von Akustikkoppler tauschten hier ihre Erfahrungen und Telefonnummern aus.
Vor sachkundigen Zuschauern zeigte Jim Butterfield die Fähigkeiten des Commodore Plus 4. Neben 4 festeingebauten Programmen, die jeder beim Kauf selbst aussuchen kann, verfügt der Plus 4 über einen 60-KB-Speicher. Die Workshops, die auf der Messe durchgeführt wurden, waren immer gut besucht. So wie die Messe selber in 3 Tagen von fast 18 000 Besuchern besucht wurde. Die 4. Internationale Commodore Fachausstellung wurde zum Erfolg für Commodore. Die Aussteller und Besucher waren zufrieden.

„Caissa"-Schachprogramm – eine deutsche Computer-Software

Das Schachprogramm „Caissa" stellt den Beweis für ein deutsches Computerprogramm dar. Als der Autor, der 27jährige Kieler Physikstudent Friedhelm Wrensch, der Firma Dynamics eine Arbeitskopie des Schachprogramms „Caissa" zur Vermarktung anbot, orientierte man sich an bestehenden Schachprogrammen, untersuchte diese auf Stärken und Schwächen und beschloß das bis zu diesem Zeitpunkt bestehende Programm um viele Funktionen zu erweitern. In einem Pflichtenheft wurde dem Autor klar umrissen vorgegeben, welche Änderungen nötig seien, um der Aussage „eines der stärksten Schachprogramme für den C64" gerecht zu werden. In Zusammenarbeit mit Schachspielern wurde das Programm auf seine Stärke hin untersucht, selbstverständlich spielte es auch gegen auf dem Markt befindliche Programme,um die Spielstärke objektivieren zu können. Nach Abschluß all dieser Tests wurde dann das Programm zum Verkauf freigegeben.
Caissa, die Göttin des Schachs, ist eines der stärksten Schachprogramme für den Commodore C64. Sich voll nach den Regeln des Turnierschachs richtend, hat es die Fähigkeit, die Bedenkzeit des Gegners zu nutzen, um weiter zu rechnen. Dabei kann es bis zu 19 Halbzüge im Voraus berechnen. Die Anzahl dieser Züge wird ebenso angezeigt, wie die der bereits ausgeführten. Sowohl die Neueingabe als auch eine Änderung von Stellungen (selbst während des Spielens) ist möglich. Die Lösung von Schachproblemen (Matt in ein bis zehn Zügen) ist ebenso vorgesehen, wie eine Autoplayfunktion. Zu den weiteren Vorzügen des Programms gehören: Automatische Bauernumwandlung (Unterumwandlungen möglich), Rücknahme von Zügen und Zugvorschlägen. Interessante Partien lassen sich auf Kassette oder Diskette abspeichern. *Gesehen bei: Dynamics marketing GmbH, 2000 Hamburg 1.*

„CAISSA"



„FORMELSAMMLUNG 64"

In Zusammenhang mit dem Girardet Verlag präsentiert DYNAMICS eine Neuheit auf dem Softwaremarkt: *Naturwissenschaftliche Formelsammlungen für den Commodore C64*

Mit über 700 Formeln, 3 Tabellenwerken, integriertem wissenschaftlichen Taschenrechner, Schlagwortregister, automatischer Datensicherung, optischer Benutzerführung, voller Menuesteuerung und automatischer Datenübernahme zu weiteren Formelsammlungen seien hier nur einige der wichtigsten Programminhalte aufgeführt.
Dem Anwender wird mit diesem Programm die Möglichkeit gegeben, in vorhandene Formeln Werte einzusetzen. Das Ergebnis ist abspeicherbar und kann in weiteren Formeln und Formelsammlungen verwendet werden. Ein spezieller Zeichensatz unterstützt die Bildschirmdarstellung von Sonderzeichen.
Um eine schnelle Orientierung und gezielten Programmzugriff zu ermöglichen, ist eine Formelsammlung des Girardet Verlages beigelegt.
Folgende Formelsammlungen werden in Kürze lieferbar sein:
Physik, Mathematik I & II, Chemie, Elektronik, Statik, Elektrotechnik, Kfz-Technik, Sanitärhandwerk, Metallgewerbe, Bau-Holzhandwerk.
Programm auf Diskette.

Unser absolutes Top-Programm in dieser Ausgabe.

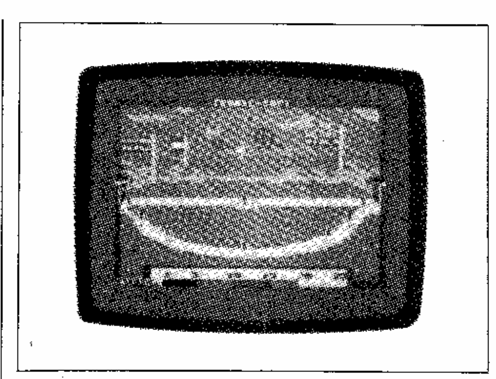
Auf vielfachen Wunsch hat sich die Redaktion entschieden, das Programm „Skeet", bekannt aus unserem Kassettenservice, als „Tontaubenschießen-Version", abzudrucken.

Verwendet wurde zur Erstellung des Programmes ein neues Verfahren unseres Hauses, um das abgedruckte Listing so kurz wie möglich zu gestalten. Der Vorteil liegt auf der Hand. Trotz gehobener Spielqualität ist das hier abgedruckte Listing von „Skeet" relativ kurz (Das komplette Programm wurde um die Hälfte reduziert!!).

Trotzdem bitten wir Sie, liebe Leser, bei der Eingabe des Listings, im besonderen der Datazeilen, sehr sorgsam vorzugehen. Die häufigste Ursache von Störungen im Programm sind nun einmal unterlaufene Fehler bei der Eingabe.

Das Programm „Skeet" ist dem realistischen Tontaubenschießen nachempfunden. Von einem Katapult geschleuderte Tontauben müssen reaktionsschnell getroffen werden. In diesem Spiel kann also jeder seine eigene Meisterschaft austragen. Eine Supergrafik und ein hervorragender Bewegungsablauf zeichnen „Skeet" besonders aus.

Hinweis: Unser neues Verfahren zur Erstellung kurzer und guter Programme wie „Skeet", „Projekt" und „Spiders" ermöglicht es uns, Ihnen, liebe Leser, in jeder Ausgabe eine Spielversion der Spitzenklasse für den VC 64 vorzustellen und abzudrucken. Wir wünschen Ihnen schon jetzt: gute Unterhaltung!



```
1 REM ************************
2 REM * TRONIC-SOFT PRESENTS *
3 REM *     S K E E T        *
4 REM * CREATED  BY F.BRALL  *
5 REM * GRAPHICS BY R.BECK   *
6 REM ************************
7 :
8 REM CODEWORT IST: COMPUTRONIC
9 :
10 POKE 56,128:REM BASIC-$8000
30 :
40 POKE 53281,1
50 PRINT"▒▒▒ ************* TRONIC-SOFT ************ "
60 PRINT"▒ ▒            S K E E T     "
70 PRINT"▒            ______________________"
80 PRINT"▒▒▒    CREATED  BY: FRANK BRALL"
90 PRINT"▒▒    GRAPHICS BY: RAINER BECK"
100 PRINT"▒▒▒▒______________________________________________"
110 PRINT"▒  COPYRIGHT BY TRONIC-VERLAG (10.1984)"
120 PRINT"▒_____________________________________________"
130 FORT=1 TO 17:FORI=0 TO 50:POKE 53280,I:NEXTI:POKE 53281,T:NEXTT
140 INPUT"▒   CODEWORT ";C$
150 IFC$<>"COMPUTRONIC"THEN 50
160 FORI=49984 TO 50046:READ DA:POKE I,DA:NEXTI:REM TAUBE      = BLOCK 13
170 FORI=50048 TO 50110:READ DA:POKE I,DA:NEXTI:REM ZIELKREUZ  = BLOCK 14
180 FORI=50112 TO 50174:READ DA:POKE I,DA:NEXTI:REM EXPLOSION1= BLOCK 15
190 FORI=49856 TO 49918:READ DA:POKE I,DA:NEXTI:REM EXPLOSION2= BLOCK 11
200 :
210 GOSUB1660:POKE 53280,6
220 POKE 53240,14:POKE 53241,13:POKE 53242,13:POKE 53243,15:POKE 53244,11
230 POKE 53287,0:POKE 53288,2:POKE 53289,2:REM COLOR
260 :
270 REM --- BASIC TEIL ---
280 :
290 POKE 838,12:REM TIME
```

```
300 POKE 56325,25 :REM IRQ TIMER
310 A1=53192  :T=0
320 :
330 POKE 832,37:POKE 833,0:POKE 53249,146
340 GOSUB 1360
350 SYS 33000:REM WAIT
360 SYS 33003:SYS 33006
370 :
380 POKE 832,37:POKE 833,0:POKE 53249,146
390 GOSUB 1310
400 SYS 33000:REM WAIT
410 SYS 33003:SYS 33012
420 :
430 POKE 832,48:POKE 833,0:POKE 53249,158
440 GOSUB 1310
450 SYS 33000:REM WAIT
460 SYS 33003:SYS 33006
470 :
480 POKE 832,48:POKE 833,0:POKE 53249,158
490 GOSUB 1310
500 SYS 33000:REM WAIT
510 SYS 33003:SYS 33009
520 :
530 POKE 832,48:POKE 833,0:POKE 53249,158
540 GOSUB 1310
550 SYS 33000:REM WAIT
560 SYS 33003:SYS 33012
570 :
580 POKE 832,83:POKE 833,0:POKE 53249,176
590 GOSUB 1310
600 SYS 33000:REM WAIT
610 SYS 33003:SYS 33006
620 :
630 POKE 832,83:POKE 833,0:POKE 53249,176
640 GOSUB 1310
650 SYS 33000:REM WAIT
660 SYS 33003:SYS 33009
670 :
680 POKE 832,83:POKE 833,0:POKE 53249,176
690 GOSUB 1310
700 SYS 33000:REM WAIT
710 SYS 33003:SYS 33012
720 :
730 POKE 832,172:POKE 833,0:POKE 53249,186
740 GOSUB 1310
750 SYS 33000:REM WAIT
760 SYS 33003:SYS 33006
770 :
780 POKE 832,172:POKE 833,0:POKE 53249,186
790 GOSUB 1310
800 SYS 33000:REM WAIT
810 SYS 33003:SYS 33009
820 :
830 POKE 832,7  :POKE 833,1:POKE 53249,176
840 GOSUB 1310
850 SYS 33000:REM WAIT
860 SYS 33003:SYS 33006
870 :
880 POKE 832,7  :POKE 833,1:POKE 53249,176
890 GOSUB 1310
900 SYS 33000:REM WAIT
910 SYS 33003:SYS 33009
920 :
930 POKE 832,7  :POKE 833,1:POKE 53249,176
940 GOSUB 1310
950 SYS 33000:REM WAIT
960 SYS 33003:SYS 33012
970 :
980 POKE 832,36 :POKE 833,1:POKE 53249,158
990 GOSUB 1310
1000 SYS 33000:REM WAIT
1010 SYS 33003:SYS 33006
1020 :
1030 POKE 832,36 :POKE 833,1:POKE 53249,158
1040 GOSUB 1310
1050 SYS 33000:REM WAIT
1060 SYS 33003:SYS 33009
1070 :
1080 POKE 832,36 :POKE 833,1:POKE 53249,158
1090 GOSUB 1310
1100 SYS 33000:REM WAIT
1110 SYS 33003:SYS 33012
1120 :
1130 POKE 832,47 :POKE 833,1:POKE 53249,146
1140 GOSUB 1310
1150 SYS 33000:REM WAIT
1160 SYS 33003:SYS 33012
1170 :
1180 POKE 832,171 :POKE 833,0:POKE 53249,146
1190 GOSUB 1310
1200 SYS 33000:REM WAIT
1210 SYS 33003:SYS 33009
1220 :
1230 POKE 832,171 :POKE 833,0:POKE 53249,146
1240 GOSUB 1310
1250 SYS 33000:REM WAIT
1260 SYS 33003:SYS 33006
1270 :
1280 GOSUB1310:SYS33000:REM WAIT
1290 FORI=53191 TO 53191+25:IF PEEK(I)=160THEN
     POKEI,102
1300 NEXTI:  RUN270
1310 :
1320 REM *** TREFFER ANZEIGEN ***
1330 :
```

```
1340 IF PEEK(53241)<>13 THEN T=T+1:POKE A1,160 :A1=A1+1
1350 IF PEEK(53242)<>13 THEN T=T+1:POKE A1,160:A1=A1+1
1360 :
1370 REM *** WINKEL-ZUFALL ***
1380 :
1390 W1=INT(RND(1)*4 )+2:W1=W1*2
1400 POKE 837,W1:RETURN
1410 :
1430 REM TONTAUBE
1440 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1450 DATA0,0,0,0,0,0,0,0,0,0,14,0,0,63,128
1460 DATA0,255,224,0,0,0,0,0,0,0,0,0,0,0,0,0
1470 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
1480 REM ZIELKREUZ
1490 DATA0,255,128,1,8,64,2,8,32,4,8,16,8,8,8,16
1500 DATA8,4,32,62,2,32,73,2,32,136,130,32,136,130,63,255
1510 DATA254,32,136,130,32,136,130,32,73,2,32,62,2,16,8,4,8
1520 DATA8,8,4,8,16,2,8,32,1,8,64,0,255,128
1530 REM EXPLOSION 1
1540 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1550 DATA8,0,4,145,0,26,132,0,2,74,128,54,48,0,25,54
1560 DATA64,2,108,128,43,73,0,4,49,64,9,6,0,8,18,0,1
1570 DATA8,0,0,0,0,0,0,0,0,0,0,0,0
1580 REM EXPLOSION 2
1590 DATA0,0,0,0,0,4,0,128,0,0,32,0,68,80,16
1600 DATA0,0,2,0,0,0,0,18,0,0,0,48,64,0,16,0
1610 DATA0,0,0,1,0,2,0,1,0,0,0,0,0,12,128,8,0
1620 DATA0,32,0,4,0,0,0,0,0,8,0,0,0,0
1630
1650 :
1660 REM VIDEOCONTROLLER UMSTELLEN
1670 REM SPRITES AB       49152 = $C000
1680 REM SPRITESPOINTER  53240 = $CFF8
1690 REM VIDEOSCREEN AB  52224 = $CC00
1700 REM ZEICHENSATZ AB  53248 = $D000
1710 REM
1720 FORI= 32768 TO 32900 :READ DA:POKEI,DA:NEXTI:SYS 32768
1730 REM *** MASCHINENPROGRAMM ***
1740 DATA 76,6,128,76,73,128,120,173,0,221,41,252,141,0,221,169,52
1750 DATA 141,24,208,169,204,141,136,2,169,97,141,17,3,169,128,141,18
1760 DATA 3,160,0,132,3,169,208,133,4,162,16,169,51,133,1,177,3
1770 DATA 72,169,48,133,1,104,145,3,200,208,239,230,4,202,208,234,169
1780 DATA 55,133,1,88,96,120,165,1,72,32,253,174,32,235,183,169,48
1790 DATA 133,1,160,0,138,145,20,104,133,1,88,96,165,20,72,165,21
1800 DATA 72,32,247,183,165,1,72,169,52,120,133,1,160,0,177,20,168
1810 DATA 104,133,1,88,104,133,21,104,133,20,76,162,179,255
1820 REM *** ZEICHENSATZ AENDERN ***
1830 DATA  33 ,0,0,0,0,0,0,0,0
1840 DATA  77 ,255,255,255,255,255,255,255,31
1850 DATA  176 ,255,255,255,255,127,31,7,0
1860 DATA  177 ,191,159,135,129,128,128,128,0
1870 DATA  178 ,0,1,3,15,31,127,255,255
1880 DATA  179 ,0,248,252,252,254,255,255,255
1890 DATA  180 ,255,255,255,255,255,255,252,0
1900 DATA  181 ,175,218,173,251,150,213,183,173
1910 DATA  182 ,171,245,106,183,101,223,169,101
1920 DATA  183 ,255,189,247,255,181,255,255,187
1930 DATA  184 ,255,239,191,255,235,255,255,247
```

```
1940 DATA  185 ,213,234,253,255,255,255,255,255
1950 DATA  186 ,205,170,117,176,237,254,255,255
1960 DATA  187 ,179,85,174,13,183,127,255,255
1970 DATA  188 ,171,87,191,255,255,255,255,255
1980 DATA  210 ,255,255,255,255,255,255,0,0
1990 DATA -1
2000 READ ZE
2010 IF ZE<>-1 THEN FORI=0TO7:READ DA:SYS32771,(53248+(ZE*8)+I),DA:NEXT
2020 IF ZE<>-1 THEN  2000
2030 :
2040 PRINT"                         TRONIC-SOFT                   ";
2050 PRINT"               :;                          9   :655  ";
2060 PRINT"    :565:67                      465        89:;  ";
2070 PRINT"                                   :5;:;  :;        ";
2080 PRINT"         .           :;              WEHRE          ";
2090 PRINT" ===**                                  *TAL*         ";
2100 PRINT" =====                                   |           ";
2110 PRINT"       |     |      |                       | 4      ";
2120 PRINT"  ;:84 |   4 |                              |:57     ";
2130 PRINT"  7665;|  76;  687           466   :6  :55656:  ";
2140 PRINT"   /555 :;6;:6;99;;9:;;:65;:::;#66;:66\     ";
2150 PRINT"  ,5;;              .    .        .      99:6 ,";
2160 PRINT" |  8        .                          .   9 | ";
2170 PRINT"   I:                       .I:              I:   ";
2180 PRINT"   . .   .                                  .    ";
2190 PRINT"     1 I .                           . I:       ";
2200 PRINT"  .      . ...                         .   .     ";
2210 PRINT"                I .                      I        .  ";
2220 PRINT"        3   .  1          .  ..             .  2     ";
2230 PRINT"   . .                      I                    .   ";
2240 PRINT"     .  .                  . .:               .   .   ";
2250 PRINT"                                                    ";
2260 PRINT"   .                                           .  .  ";
2270 PRINT"         1       5       10      15      20      25        ";
2280 PRINT"TREFFER  .....  .....  .....  .....  .....         ";
2290 POKE 53223,160:POKE 56295,2
2300 POKE 53281,1
50000 FORI= 33000 TO 34283 :READ DA:POKEI,DA:NEXTI
60000 DATA 76,130,131,76,247,128,76,239,129,76,3,131,76,170,131,120,169
60001 DATA 27,141,20,3,169,129,141,21,3,169,0,141,72,3,169,2,141
60002 DATA 71,3,88,96,120,169,49,141,20,3,169,234,141,21,3,88,96
60003 DATA 169,224,141,2,220,173,0,220,141,68,3,169,255,141,2,220,173
60004 DATA 68,3,41,1,208,3,32,147,129,173,68,3,41,2,208,3,32
60005 DATA 151,129,173,68,3,41,4,208,3,32,89,129,173,68,3,41,8
60006 DATA 208,3,32,117,129,76,155,129,76,49,234,173,65,3,208,5,173
60007 DATA 64,3,240,17,56,173,64,3,233,1,141,64,3,173,65,3,233
60008 DATA 0,141,65,3,96,173,65,3,240,7,173,64,3,201,80,16,243
60009 DATA 24,173,64,3,105,1,141,64,3,173,65,3,105,0,141,65,3
60010 DATA 96,206,1,208,96,238,1,208,96,173,64,3,141,0,208,173,65
60011 DATA 3,240,11,173,16,208,9,1,141,16,208,76,86,129,173,16,208
60012 DATA 41,254,141,16,208,76,86,129,173,64,3,141,0,208,173,65,3
60013 DATA 240,9,173,16,208,9,1,141,16,208,96,173,16,208,41,254,141
60014 DATA 16,208,96,72,138,72,152,72,174,70,3,160,150,136,208,253,202
60015 DATA 208,248,104,168,104,170,104,96,169,0,141,67,3,169,50,141,72
60016 DATA 3,169,40,141,66,3,169,130,141,3,208,172,69,3,32,114,130
60017 DATA 173,72,3,208,6,32,172,130,76,23,130,206,72,3,24,173,66
60018 DATA 3,105,1,141,66,3,173,67,3,105,0,141,67,3,136,208,6
```

```
60019 DATA 172,69,3,206,3,208,32,114,130,173,249,207,201,15,208,15,206
60020 DATA 78,3,208,10,169,11,141,249,207,169,20,141,78,3,173,249,207
60021 DATA 201,11,208,13,206,78,3,208,8,173,21,208,41,253,141,21,208
60022 DATA 32,217,129,173,67,3,201,1,208,161,173,66,3,201,100,208,154
60023 DATA 76,14,129,173,66,3,141,2,208,173,67,3,240,9,173,16,208
60024 DATA 9,2,141,16,208,96,173,16,208,41,253,141,16,208,96,173,73
60025 DATA 3,141,4,208,173,74,3,240,9,173,16,208,9,4,141,16,208
60026 DATA 96,173,16,208,41,251,141,16,208,96,120,169,224,141,2,220,173
60027 DATA 0,220,141,68,3,169,255,141,2,220,173,68,3,41,16,240,2
60028 DATA 88,96,173,71,3,240,249,169,10,141,1,212,169,0,141,4,212
60029 DATA 169,15,141,24,212,169,9,141,5,212,169,131,141,4,212,32,240
60030 DATA 132,173,75,3,208,10,169,30,141,72,3,206,71,3,88,96,169
60031 DATA 15,141,249,207,169,20,141,78,3,76,236,130,169,50,141,72,3
60032 DATA 169,1,141,67,3,169,60,141,66,3,169,130,141,3,208,172,69
60033 DATA 3,32,114,130,173,72,3,208,6,32,172,130,76,43,131,206,72
60034 DATA 3,56,173,66,3,233,1,141,66,3,173,67,3,233,0,141,67
60035 DATA 3,136,208,6,172,69,3,206,3,208,32,114,130,173,249,207,201
60036 DATA 15,208,15,206,78,3,208,10,169,11,141,249,207,169,20,141,78
60037 DATA 3,173,249,207,201,11,208,13,206,78,3,208,8,173,21,208,41
60038 DATA 253,141,21,208,32,217,129,173,67,3,208,163,173,66,3,208,158
60039 DATA 76,14,129,169,13,141,249,207,141,250,207,169,15,141,21,208,32
60040 DATA 188,129,169,224,141,2,220,173,0,220,141,68,3,169,255,141,2
60041 DATA 220,173,68,3,41,16,208,233,96,169,50,141,72,3,169,0,141
60042 DATA 67,3,169,40,141,66,3,169,1,141,74,3,169,60,141,73,3
60043 DATA 169,130,141,3,208,141,5,208,172,69,3,32,114,130,32,143,130
60044 DATA 173,72,3,208,6,32,132,132,76,226,131,206,72,3,24,173,66
60045 DATA 3,105,1,141,66,3,173,67,3,105,0,141,67,3,56,173,73
60046 DATA 3,233,1,141,73,3,173,74,3,233,0,141,74,3,136,208,9
60047 DATA 172,69,3,206,3,208,206,5,208,32,114,130,32,143,130,173,249
60048 DATA 207,201,15,208,15,206,78,3,208,10,169,11,141,249,207,169,20
60049 DATA 141,78,3,173,249,207,201,11,208,13,206,78,3,208,8,173,21
60050 DATA 208,41,253,141,21,208,173,250,207,201,15,208,15,206,79,3,208
60051 DATA 10,169,11,141,250,207,169,20,141,79,3,173,250,207,201,11,208
60052 DATA 13,206,79,3,208,8,173,21,208,41,251,141,21,208,32,217,129
60053 DATA 173,67,3,201,1,240,3,76,212,131,173,66,3,201,100,240,3
60054 DATA 76,212,131,76,14,129,120,169,224,141,2,220,173,0,220,141,68
60055 DATA 3,169,255,141,2,220,173,68,3,41,16,240,2,88,96,173,71
60056 DATA 3,240,249,169,10,141,1,212,169,0,141,4,212,169,15,141,24
60057 DATA 212,169,9,141,5,212,169,131,141,4,212,32,68,133,173,75,3
60058 DATA 240,34,173,75,3,41,1,240,10,169,15,141,249,207,169,20,141
60059 DATA 78,3,173,75,3,41,2,240,10,169,15,141,250,207,169,20,141
60060 DATA 79,3,169,30,141,72,3,206,71,3,88,96,169,0,141,75,3
60061 DATA 56,173,66,3,233,12,141,76,3,173,67,3,233,0,141,77,3
60062 DATA 56,173,64,3,237,76,3,141,76,3,173,65,3,237,77,3,141
60063 DATA 77,3,201,0,208,38,173,76,3,41,224,208,31,24,173,1,208
60064 DATA 105,8,141,76,3,56,173,76,3,237,3,208,141,76,3,173,76
60065 DATA 3,41,224,208,5,169,1,141,75,3,96,169,0,141,75,3,56
60066 DATA 173,66,3,233,12,141,76,3,173,67,3,233,0,141,77,3,56
60067 DATA 173,64,3,237,76,3,141,76,3,173,65,3,237,77,3,141,77
60068 DATA 3,201,0,208,38,173,76,3,41,224,208,31,24,173,1,208,105
60069 DATA 8,141,76,3,56,173,76,3,237,3,208,141,76,3,173,76,3
60070 DATA 41,224,208,5,169,1,141,75,3,234,56,173,73,3,233,12,141
60071 DATA 76,3,173,74,3,233,0,141,77,3,56,173,64,3,237,76,3
60072 DATA 141,76,3,173,65,3,237,77,3,141,77,3,201,0,208,41,173
60073 DATA 76,3,41,240,208,34,24,173,1,208,105,8,141,76,3,56,173
60074 DATA 76,3,237,5,208,141,76,3,173,76,3,41,240,208,8,173,75
60075 DATA 3,9,2,141,75,3,96,127,255
60080 RETURN
```

Ein Sprite- und Zeichen-Designer mit viel Komfort für den C-64. Mit dem hier vorgestellten Programm lassen sich Multicolor- sowie normale Sprites auf einfachste Weise erstellen.

Da außerdem auch noch der Zeichensatz des Commodore 64 geändert werden kann, ist dieses Programm jedem Programmierer zu empfehlen. Das Programm ist mit einem Generator ausgestattet, welcher ein Basic-Unterprogramm erzeugt, das in Spielen verwendet werden kann. Dieses Unterprogramm besteht aus einigen Data-Zeilen, die eine Maschinenroutine, Sprites und die neu definierten Zeichen enthalten.
Durch Aufruf des Unterprogrammes wird zuerst der Video-Bereich, welcher sonst im Bereich 0 bis 16384 liegt, nach 49152 bis 65536 umgelegt. Der Bildschirm ist nun statt 1024 ab Adresse 52224 erreichbar. Nach dieser Umbelegung besteht die Möglichkeit 48 Sprites zu definieren, ohne den Basic-Bereich zu überschreiben. Die Sprites müssen nun ab Adresse 49152 abgelegt werden. Die Block-Nummern (Spritepointer) sind ab der Adresse 53240 erreichbar. Zu beachten ist also, das Block 0 ab Adresse 49152 und nicht ab Adresse 0 vom Controller gelesen wird.
Außer dieser Umbelegung wird der Zeichensatz in den Ram-Bereich $D000–$DFFF kopiert und nachträglich um die neu definierten Zeichen geändert. Ebenso werden die definierten Sprites in die entsprechenden Blöcke gelesen.

Start des Designers
Nach dem Start des Programmes mit „RUN" kommt man in ein Hauptmenue, welches folgende Möglichkeiten bietet:
Taste F1 Einfarbige Sprites definieren bzw. ändern
Taste F3 Multicolor-Sprites definieren bzw. ändern
Taste F5 Zeichensatz umdefinieren
Taste F7 Sprites und Zeichensatz werden abgespeichert
Taste F2 Sprites und Zeichensatz werden eingeladen
Taste F4 Ein Unterprogramm mit den Sprites und den definierten Zeichen wird erstellt.
Nach drücken der jeweiligen Funktions-Taste kommt man in das entsprechende Unterprogramm. Wie diese im einzelnen ablaufen, wird in den nächsten Abschnitten beschrieben.

Einfarbige Sprites
Nach Aufruf dieses Programmteiles erscheint links ein 24 x 21 großes Fenster (Editier-Fenster), indem mit den Cursor-Tasten beliebig umhergefahren werden kann. Mit der Taste „1" können Punkte gesetzt und mit der Taste „<–" gelöscht werden. Unabhängig von diesem Fenster stehen auf der rechten Bildseite einige Befehle, welche die Definition erheblich erleichtern können. Im einzelnen leisten diese Befehle folgendes:
„+"-und „–"-Befehl
Unabhängig von dem linken Editier-Fenster wird am rechten unteren Rand ein Sprite dargestellt, wir nennen dieses Sprite-Fenster. Über diesem Sprite-Fenster wird angezeigt, welcher Block gerade im Sprite-Fenster sichtbar ist. Mit den Tasten „+" kann man nun die Block-Nr. erhöhen und somit ein anderes Sprite sichtbar machen. Mit dem Befehl „–" läßt sich die Block-Nr. wieder erniedrigen.
„H"- und „V"-Befehl
Mit „H" kann man das Sprite-Fenster horizontal vergrößern und wieder verkleinern. Mit „V" ist die vertikale Größe umschaltbar.
E (Editieren)
Dieser Befehl übernimmt das dargestellte Sprite-Fenster in das große Editier-Fenster. Zu beachten ist, daß das Editier-Fenster nicht vorher gelöscht wird und deshalb schon gesetzte Punkte erhalten bleiben.
C (Clear)
Das Editier-Fenster wird gelöscht.
= (Definieren)
Das mit dem Editier-Fenster erstellte Sprite wird von dem angezeigten Sprite-Fenster übernommen.
I (Invertieren)
Das Editier-Fenster wird invertiert.
R (Rotieren)
Das Sprite wird um eine Achse im Uhrzeigersinn gedreht. Da Sprites nicht ganz quadratisch sind, gehen einige Punkte verloren.
X (Spiegeln)
Das Sprite wird um die vertikale Achse gedreht. Möchte man beispielsweise ein Männchen zeichnen, so braucht man dieses nur in eine Richtung zu zeichnen und dreht es dann mit diesem Befehl um.
Y (Spiegeln)
Entspricht dem X-Befehl, jedoch wird hier um die horizontale Achse gedreht.
F1 (Color Ground)
Mit dieser Taste läßt sich die Hintergrundfarbe des Bildschirmes wechseln.
F3 (Color Sprite)
Mit dieser Taste läßt sich die Farbe des Sprites ändern.
>↑< Mit dieser Taste kommt man wieder ins Hauptmenue.

Multicolor-Sprites
Nach Aufruf dieses Programmteiles erscheint wie bei den einfarbigen Sprites links ein Editier-Fenster. Dieses ist jedoch in diesem Fall nur 12 x 12 Felder groß, da jeder Punkt 3 Farben annehmen kann. Zu bedenken ist, daß später jeder Punkt in doppelter Breite dargestellt wird. Mit folgenden Befehlen kann gearbeitet werden:
Taste „1" setzt Punkt mit Farbe 1
Taste „2" setzt Punkt mit Farbe 2
Taste „3" setzt Punkt mit Farbe 3
Taste „<–" löscht Punkt, also Hintergrundfarbe
Taste „F1" wählt Hintergrundfarbe
Taste „F3" wählt Farbe 1
Taste „F5" wählt Farbe 2
Taste „F7" wählt Farbe 3
Die Befehle X, Y, C, =, E, +, –, H, V haben die gleiche Funktion wie bei den einfarbigen Sprites und werden deshalb nicht mehr beschrieben.

S (SWAP)
Zwei Sprites können ausgetauscht werden. Beispielsweise kann man Spriteblock 3 und Spriteblock 40 mit folgender Tastenfolge austauschen:
S
3 (Return)
40 (Return)

„T"- und „R"-Befehl
Neben dem normalen Sprite-Fenster gibt es noch ein Vierer-Fenster. Dieses befindet sich etwa in der unteren Mitte und zeigt 4 hintereinanderfolgende Blöcke gleichzeitig an. Man hat so die Möglichkeit, mehrere Sprites für eine Figur zu definieren. Mit der Taste „T" kann man jeweils um 4 Blöcke vorspringen und mit Taste „R" jeweils um 4 Blöcke zurückspringen.
D (Data in)
Mit der Hilfe dieses Befehles kann in den angewählten Block ein Sprite durch die dezimale Eingabe von 63 Zahlen definiert werden. Möchte man nur sehen, welche Zahlen in dem Block definiert sind, so genügt das einmalige Drücken der Taste D sowie das Weiterzählen mit „Return". Möchte man während der Eingabe um eine

Zahl zurück, so geschied dies mit dem Befehl „–". Mit „★" kann man die Eingabe abbrechen.

Zeichensatz ändern

Um beispielsweise schöne Spiele programmieren zu können, ist auf die Definition neuer Zeichen kaum noch zu verzichten. Gegenüber der hochauflösenden Grafik hat die Definition von Sonderzeichen folgende Vorteile:

- Einfacher zu definieren.
- Wesentlich weniger Speicherplatz wird benötigt.
- Wesentlich schneller im Spiel zu verändern (Print/Poke).

Der hier vorgestellte Zeichen-Designer ist an Luxus wohl kaum noch zu übertreffen. Es lassen sich bis maximal 512 Zeichen definieren.

Nach dem Start erscheinen links oben gleich 4 Editier-Fenster, die bündig aneinandersitzen, es ist so möglich, gleich mehrere Zeichen für eine Figur zu definieren.

Mit „1" wird ein Punkt gesetzt und mit „<–" wieder gelöscht.

Mit „C" werden alle Editier-Fenster gleichzeitig gelöscht.

Die Befehle X, Y, R, I entsprechen den schon beschriebenen Funktionen. Zusätzlich hat man nach der Eingabe des Befehles die Wahl, ob man nur ein bestimmtes Fenster oder auch gleich alle behandeln möchte. Um beispielsweise Fenster 2 zu rotieren, geben Sie R 2 (Return) ein. Um alle zu rotieren, gibt man RA (Return) ein.

E (Editieren)

Nach Eingabe dieses Befehles fragt der Computer erst nach dem Fenster (1–4) und dann nach dem Zeichen. Jedem Zeichen ist eine Zahl von 0 bis 255 zugeordnet, welche leicht durch die rechts angezeigte Zeichen-Tabelle erkennbar ist.

= (Definieren)

Wie bei dem E-Befehl wird erst nach dem Fenster und dann nach dem Zeichen gefragt. Das erstellte Zeichen wird dann an die gewünschte Stelle gelegt.

T (Table/Zeichensatz wählen)

Mit diesem Befehl kann man zwischen zwei Zeichensätzen umschalten. Entweder Groß-/Kleinschrift oder Großschrift/Grafik.

Save Sprites und Zeichensatz

Durch Drücken der Funktionstaste F7 wird der definierte Zeichensatz sowie alle 48 Sprite-Blöcke auf Diskette geschrieben. Ersetzt man in Zeile 38040 die Zahl 8 durch eine 1, so kann man auch auf Kassette saven. Da dies jedoch schon auf Diskette bis zu 4 Minuten dauern kann, ist Kassette weniger sinnvoll.

Load Sprites und Zeichensatz

Dies ist das Gegenstück zum Save-Befehl. Sprites und Zeichensatz können eingeladen und erweitert oder editiert werden. Um mit Kassette arbeiten zu können, muß in Zeile 39040 die Zahl 8 durch eine 1 ersetzt werden.

Programm-Generator

Sind alle Zeichen und/oder Sprites definiert und abgespeichert, so hat man mit diesem Befehl die Möglichkeit, die Sprites und neuen Zeichen in kompakter Form als Basic-Programm abzulegen.

Nach Betätigen der Taste F4 erfolgt die Frage nach Sprites von/bis. Hier muß nun die erste und letzte Block-Nummer der definierten Sprites angegeben werden. Hat man kein Sprite, sondern nur Zeichen definiert, so wählt man die erste Block-Nr. größer als die zweite.

Nach dieser Eingabe erscheint die Meldung „Wait 5 Min!", in dieser Zeit wird nun das am Anfang dieser Beschreibung erwähnte Unterprogramm erzeugt. Ist der Generierungsvorgang beendet, so steht das neue Programm im Speicher und wird automatisch ausgelistet. Neben der Verschiebung des Controllers setzt dieses Unterprogramm das Basic-Ende auf $8000 herab. Der Speicherbereich von 32901 bis 40959 ist nicht belegt und kann somit für Maschinenroutinen benutzt werden.

Bei der Erweiterung des Unterprogrammes zu einem Spiel sollte man die Zeilen von 1 bis 49998 benutzen.

Achtung!

Da das hier abgedruckte Basic-Programm ein Maschinenprogramm enthält, sollte man bei der Eingabe sehr gewissenhaft vorgehen. Besonders sollte man darauf achten, daß man beim Abtippen der Data-Zeilen keinen Fehler macht, da der Computer sonst abstürzen kann. Ebenso sollte man darauf achten, daß man die Zeilen-Nummern und REM-Zeilen so übernimmt, wie sie hier abgedruckt sind. Vor dem Start mit RUN sollte man das Programm mindestens einmal abgespeichert haben.

```
1 REM ********************  ****   ****
2 REM * GRAFIK-DESIGNER *  *      *  *
3 REM *  (C) E.REIF     *  **     ****
4 REM *   BRD  1984     *  *      * *
5 REM ********************  **** * *  *
6 REM
10 POKE 56,128:REM BASIC - $8000
15 POKE 650,128:DIM W(503)
100 FORI=32768 TO 32908:READ DA:POKE I,D
A:NEXTI:GOSUB 50000
102 RESTORE:FORI=32768 TO 32908:READ DA:
POKE I,DA:NEXTI
104 :
105 REM *** MASCHINENPROGRAMM ***
106 :
110 DATA 76,6,128,76,81,128,120,173,0,22
1,41,252,141,0,221,169,56
111 DATA 141,24,208,169,204,141,136,2,16
9,105,141,17,3,169,128,141,18
112 DATA 3,160,0,132,3,132,247,169,208,1
33,4,169,224,133,248,162,16
113 DATA 169,51,133,1,177,3,72,169,48,13
3,1,104,145,247,200,208,239
114 DATA 230,4,230,248,202,208,232,169,5
5,133,1,88,96,120,165,1,72
115 DATA 32,253,174,32,235,183,169,48,13
3,1,160,0,138,145,20,104,133
116 DATA 1,88,96,165,20,72,165,21,72,32,
247,183,165,1,72,169,52
117 DATA 120,133,1,160,0,177,20,168,104,
133,1,88,104,133,21,104,133
118 DATA 20,76,162,179,255
10120 SYS 8*4096
10130 POKE 53281,1:POKE53280,2:POKE 5326
9,0:PRINTCHR$(142);
10200 I=0:PRINT"    GRAFIK-DESIGNER ....
........(C) E.REIF "
10210 PRINT"
      "
10220 PRINT"  |F1| -->|EINFARBIGE  SPRIT
E'S|"
10230 PRINT"
      "
```

```
10240 PRINT"■ ╭──╮        ╭────────────────────────╮"
10250 PRINT"■ |F3| -->|MEHRFARBIGE SPRITE'S|"
10260 PRINT"■ ╰──╯        ╰────────────────────────╯"
10270 PRINT"■ ╭──╮        ╭────────────────────────╮"
10280 PRINT"■ |F5| -->|ZEICHENSATZ AENDERN |"
10290 PRINT"■ ╰──╯        ╰────────────────────────╯"
10295 PRINT"■ ╭──╮        ╭────────────────────────╮"
10296 PRINT"■ |F7| -->|SAVE SPRITES/ZEICHEN|"
10297 PRINT"■ ╰──╯        ╰────────────────────────╯"
10300 PRINT"■ ╭──╮        ╭────────────────────────╮"
10301 PRINT"■ |F2| -->|LOAD SPRITES/ZEICHEN|"
10302 PRINT"■ ╰──╯        ╰────────────────────────╯"
10305 PRINT"■ ╭──╮        ╭────────────────────────╮"
10306 PRINT"■ |F4| -->|PROGRAMM GENERATOR  |"
10307 PRINT"■ ╰──╯        ╰────────────────────────╯"
10400 GET E$ :T=T+1
10410 IF E$="■"THEN 11000
10420 IF E$="■"THEN 20000
10425 IF E$="■"THEN 28000
10430 IF E$="■"THEN 38000
10435 IF E$="■"THEN 39000
10438 IF E$="■"THEN 40000
10440 IF T=20 THEN T=0:I=I+1:POKE 53280,I:IF I>8  THENI=1
10450 GOTO 10400
10999 :
11000 REM *** EINFARBIGE SPRITES ***
11001 :
11011 POKE 53280,1:POKE 53281,1:PRINT"■■  EINFARBIGE SPRITES"
11012 POKE 53248,250:POKE53249,200:POKE 53276,0
11013 POKE 53269,1:POKE 53287,0:POKE53240,BL
11015 VI=52304:FA=55376:CM=7:XM=0:XP=0:YP=0:YM=0:CS=0:CG=1
11016 FORI =0 TO 20:PRINT
11020 PRINT"■■........................";:NEXTI
11034 PRINT:PRINT"■■      ↑ --> MENUE";
11035 PRINT"■■";
11040 PRINTTAB(24)"■F1 COLOR GROUND"
11050 PRINT"                              F3 COLOR SPRITE"
11060 PRINTTAB(25)"■1   SETZE PUNKT"
11070 PRINTTAB(25)"■←   CLEAR PUNKT"
11080 PRINTTAB(25)"■X   X-SPIEGELN"
11090 PRINTTAB(25)"■Y   Y-SPIEGELN"
11100 PRINTTAB(25)"■R   ROTIEREN"
11110 PRINTTAB(25)"■I   INVERTIEREN"
11120 PRINTTAB(25)"■C   LOESCHEN"
11130 PRINTTAB(25)"■=   DEFINIEREN"
11140 PRINTTAB(25)"■E   EDITIEREN"
11150 PRINT
11160 PRINTTAB(25)"■+/- WAEHLEN"
11170 PRINTTAB(25)"■H/V GROESSE"
11180 PRINT
11190 PRINTTAB(25)"■BLOCK:":POKE FA,0
11200 PRINT"■■■■■■■■■■■■■■■■■■■"TAB(31)"  "
11210 PRINT"■■■■■■■■■■■■■■■■■■■"TAB(31)BL
11300 GET E$:IF E$=""THEN 11300
11310 IF E$="+"AND BL<47 THEN BL=BL+1:POKE 53240,BL:GOTO11200
11320 IF E$="-"AND BL>0  THEN BL=BL-1:POKE 53240,BL:GOTO11200
11330 IF E$="■"AND XP<23 THEN XP=XP+1:GOTO 12000
11340 IF E$="■"AND XP>0 THEN XP=XP-1:GOTO 12000
11350 IF E$="■"AND YP<20 THEN YP=YP+1:GOTO 12000
11360 IF E$="■"AND YP>0 THEN YP=YP-1:GOTO 12000
11370 IF E$="1" THEN POKE (FA+(40*YM)+XM),6:CM=6:GOTO 11300
11380 IF E$="←" THEN POKE (FA+(40*YM)+XM),7:CM=7:GOTO 11300
11390 IF E$="C" THEN 11000
11400 IF E$="■" THEN CG=CG+1:IF CG>15 THEN CG=0
11410 IF E$="■" THEN CS=CS+1:IF CS>15 THEN CS=0
11420 IF E$="V" AND PEEK(53271)<>0 THEN POKE 53271,0:GOTO 11300
11430 IF E$="V" AND PEEK(53271)<>1 THEN POKE 53271,1:GOTO 11300
11440 IF E$="H" AND PEEK(53277)<>0 THEN POKE 53277,0:GOTO 11300
11450 IF E$="H" AND PEEK(53277)<>1 THEN POKE 53277,1:GOTO 11300
11455 IF E$="I" THEN 13000
11456 IF E$="X" THEN 14000
11457 IF E$="Y" THEN 15000
11458 IF E$="R" THEN 16000
11460 IF E$="↑" THEN 10130
11465 IF E$="=" THEN 17000
11470 IF E$="E" THEN 18000
11500 POKE 53281,CG:POKE 53287,CS:GOTO 11300
11999 :
12000 REM *** CURSOR ZEIGEN ***
12001 :
12010 POKE(FA+(40*YM)+XM),CM
12020 YM=YP:XM=XP:CM=PEEK(FA+(40*YM)+XM)
12030 POKE(FA+(40*YP)+XP),0
12040 GOTO 11300
12999 :
```

```
13000 REM *** INVERTIEREN ***
13001 :
13005 POKE(FA+(40*YP)+XP),CM
13010 FOR YH=0 TO 20:FOR XH=0 TO 23
13030 IF(PEEK(FA+(40*YH)+XH)AND15)=7 THE
N POKE (FA+(40*YH)+XH),6:GOTO 13050
13040 IF(PEEK(FA+(40*YH)+XH)AND15)=6 THE
N POKE (FA+(40*YH)+XH),7
13050 NEXTXH:NEXTYH:CM=(PEEK(FA+(40*YP)+
XP)AND15):GOTO 11300
13999 :
14000 REM *** X-SPIEGELUNG ***
14001 :
14010 Z=0:POKE (FA+(40*YP)+XP),CM
14020 FOR YH=0 TO 20:FOR XH=0 TO 23
14040 W(Z)=(PEEK(FA+(40*YH)+XH)AND15)
14050 Z=Z+1:NEXTXH:NEXTYH
14070 Z=0:FOR YH=0 TO 20
14080 FOR XH=23 TO 0 STEP-1
14090 POKE(FA+(40*YH)+XH),W(Z)
14100 Z=Z+1:NEXTXH:NEXTYH
14110 CM=(PEEK(FA+(40*YP)+XP)AND15):GOTO
 11300
14999 :
15000 REM *** Y-SPIEGELUNG ***
15001 :
15010 Z=0:POKE (FA+(40*YP)+XP),CM
15020 FOR YH=0 TO 20
15030 FOR XH=0 TO 23
15040 W(Z)=(PEEK(FA+(40*YH)+XH)AND15)
15050 Z=Z+1:NEXTXH:NEXTYH
15060 Z=0
15070 FOR YH=20 TO 0 STEP-1
15080 FOR XH=0 TO 23
15090 POKE(FA+(40*YH)+XH),W(Z)
15100 Z=Z+1:NEXTXH:NEXTYH
15110 CM=(PEEK(FA+(40*YP)+XP)AND15):GOTO
 11300
15999 :
16000 REM *** ROTIEREN ***
16001 :
16010 Z=0:POKE (FA+(40*YP)+XP),CM
16020 FOR YH=0 TO 20
16030 FOR XH=0 TO 20
16040 W(Z)=(PEEK(FA+(40*YH)+XH)AND15)
16050 Z=Z+1:NEXTXH:NEXTYH
16060 Z=0
16070 FOR XH=20TO 0 STEP-1
16080 FOR YH=0 TO 20
16090 POKE(FA+(40*YH)+XH),W(Z)
16100 Z=Z+1:NEXTYH:NEXTXH
16110 CM=(PEEK(FA+(40*YP)+XP)AND15):GOTO
 11300
16999 :
17000 REM *** DEFINIEREN ***
17001 :
17010 Z=0:POKE (FA+(40*YP)+XP),CM
17020 FOR YH=0 TO 20
17030 FOR XH=0 TO 23 STEP 8
17040 CW=0:Q=0
17050 FOR T=7 TO 0 STEP-1
17060 IF(PEEK(FA+(40*YH)+XH+Q)AND15)=6 T
HEN CW=CW+2↑T
17070 Q=Q+1:NEXTT
17080 POKE 49152+(BL*64)+Z,CW:Z=Z+1
17090 NEXTXH:NEXTYH:GOTO11300
17999 :
18000 REM *** EDITIEREN ***
18001 :
18010 K1=49152:L1=64:YH=0:XH=0:FOR Z=0 T
O 62
18020 Q=0:FOR T=7 TO 0 STEP-1
18025 IF  PEEK(K1+(BL*L1)+Z)=0 THEN T=0:
GOTO 18050
18030 IF (PEEK(K1+(BL*L1)+Z)AND2↑T)=2↑T
THEN POKE (FA+(40*YH)+XH+Q),6
18050 Q=Q+1:NEXTT
18055 XH=XH+8:IFXH=24THEN XH=0:YH=YH+1
18060 NEXT Z
18080 GOTO 11300
19999 :
20000 REM *** VIERFARBIGE SPRITES ***
20010 :
20011 POKE 53280,1:POKE 53281,1:PRINT"
 MULTICOLOR  SPRITES":POKE 53276,255
20012 POKE 53248,150:POKE53249,200:POKE
53250,174:POKE 53251,200
20013 POKE 53252,150:POKE53253,221:POKE
53254,174:POKE 53255,221
20014 POKE 53269,63:POKE53240,BL:POKE 53
241,BL+1:POKE53242,BL+2
20015 POKE 53243,BL+3:POKE 53244,BL:POKE
 53256,250:POKE53257,205
20016 VI=52304:FA=55376
20017 CM=5:XM=0:XP=0:YP=0:YM=0:CS=0:CG=1

20018 CG=PEEK(53281):C1=PEEK(53285):C3=P
EEK(53286):C2=PEEK(53287)
20019 FORI=0 TO 20:PRINT
20020 PRINT"............";
20030 NEXTI
20034 PRINT:PRINT" ↑ --> MENUE";
20035 PRINT"":PRINT
20040 PRINTTAB(13)"F1 COLOR GROUND  S S
WAP"
20050 PRINTTAB(13)"F3 COLOR 1       T B
LOCK+4"
20060 PRINTTAB(13)"F5 COLOR 2       R B
LOCK-4"
20070 PRINTTAB(13)"F7 COLOR 3       D D
ATA IN"
20080 PRINTTAB(13)"1-3 SETZE PUNKT"
20090 PRINTTAB(13)"←  CLEAR PUNKT"
20101 PRINTTAB(13)"X  X-SPIEGELN"
20102 PRINTTAB(13)"Y  Y-SPIEGELN"
20110 PRINTTAB(13)"C  LOESCHEN"
20120 PRINTTAB(13)"=  DEFINIEREN"
20130 PRINTTAB(13)"E  EDITIEREN"
20140 PRINT
20150 PRINTTAB(13)"+/- WAEHLEN"
20160 PRINTTAB(13)"H/V GROESSE"
20170 PRINT""
```

```
20180 PRINTTAB(26)"▒BLOCK:"
20190 POKE FA,0
20200 PRINT"SQQQQQQQQQQQQQQQQQQQQQQ"TAB(31)"
      "
20201 PRINT"SQQQQQQQQQQQQQQQQQQQQQQ"TAB(31)B
L
20203 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(13
)"    "
20204 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(13
)BL
20205 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(21
)"    "
20206 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(21
)BL+1
20207 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
13)"    "
20208 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
13)BL+2
20209 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
21)"    "
20210 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
21)BL+3
20300 GET E$:IF E$="" THEN 20300
20310 IF E$="+" AND BL<47 THEN 20320
20315 GOTO 20330
20320 BL=BL+1:POKE53240,BL:POKE53241,BL+
1:POKE53242,BL+2:POKE 53243,BL+3
20321 POKE 53244,BL:GOTO 20200
20330 IF E$="-" AND BL>0 THEN 20340
20335 GOTO 20350
20340 BL=BL-1:POKE53240,BL:POKE53241,BL+
1:POKE53242,BL+2:POKE 53243,BL+3
20341 POKE 53244,BL:GOTO 20200
20350 IF E$="T" AND BL<44 THEN 20360
20355 GOTO 20370
20360 BL=BL+4:POKE53240,BL:POKE53241,BL+
1:POKE53242,BL+2:POKE 53243,BL+3
20361 POKE 53244,BL:GOTO 20200
20370 IF E$="R" AND BL>3 THEN 20380
20375 GOTO 20390
20380 BL=BL-4:POKE53240,BL:POKE53241,BL+
1:POKE53242,BL+2:POKE 53243,BL+3
20381 POKE 53244,BL:GOTO 20200
20390 IF E$="▌" AND XP<11 THEN XP=XP+1:G
OTO 21000
20395 IF E$="▐" AND XP>0 THEN XP=XP-1:GO
TO21000
20400 IF E$="▩" AND YP<20 THEN YP=YP+1:G
OTO 21000
20405 IF E$="コ" AND YP>0 THEN YP=YP-1:GO
TO 21000
20410 IF E$="1" THEN POKE (VI+(40*YM)+XM
),177:GOTO 20300
20420 IF E$="2" THEN POKE (VI+(40*YM)+XM
),178:GOTO 20300
20430 IF E$="3" THEN POKE (VI+(40*YM)+XM
),179:GOTO 20300
20440 IF E$="←" THEN POKE (VI+(40*YM)+XM
),174:GOTO 20300
20450 IF E$="V" AND PEEK(53271)<>0 THEN
POKE 53271,0 :GOTO 20300
20460 IF E$="V" AND PEEK(53271)<>16 THEN
 POKE 53271,16:GOTO 20300
20470 IF E$="H" AND PEEK(53277)<>0 THEN
POKE 53277,0:GOTO 20300
20480 IF E$="H" AND PEEK(53277)<>16 THEN
 POKE 53277,16:GOTO 20300
20490 IF E$="↑" THEN 10130
20500 IF E$="C" THEN 20000
20510 IF E$="■" THEN CG=CG+1:IF CG>15 TH
EN CG=0
20520 IF E$="▀" THEN C1=C1+1:IF C1>15 TH
EN C1=0
20530 IF E$="▐" THEN C2=C2+1:IF C2>15 TH
EN C2=0
20540 IF E$="▌" THEN C3=C3+1:IF C3>15 TH
EN C3=0
20550 IF E$="X" THEN 22000
20560 IF E$="Y" THEN 23000
20570 IF E$="=" THEN 24000
20580 IF E$="E" THEN 25000
20590 IF E$="S" THEN 26000
20595 IF E$="D" THEN 27000
20997 POKE 53281,CG:POKE 53285,C1:POKE 5
3286,C3:POKE 53287,C2:POKE 53288,C2
20998 POKE 53289,C2:POKE 53290,C2:POKE 5
3291,C2
20999 GOTO20300
21000 :
21010 REM *** CURSOR ZEIGEN ***
21011 :
21020 POKE (FA+(40*YM)+XM),CM
21030 YM=YP:XM=XP:CM=PEEK(FA+(40*YM)+XM)

21040 POKE (FA+(40*YP)+XP),0
21050 GOTO 20300
21999 :
22000 REM *** X-SPIEGELUNG ***
22001 :
22020 Z=0:FOR YH=0 TO 20
22030 FOR XH=0 TO 11
22040 W(Z)=PEEK(VI+(40*YH)+XH)
22050 Z=Z+1:NEXTXH:NEXTYH
22060 Z=0
22070 FOR YH=0 TO 20
22080 FOR XH=11 TO 0 STEP-1
22090 POKE(VI+(40*YH)+XH),W(Z)
22100 Z=Z+1:NEXTXH:NEXTYH
22110 GOTO 20300
22999 :
23000 REM *** Y-SPIEGELUNG ***
23001 :
23010 Z=0
23020 FOR YH=0 TO 20
23030 FOR XH=0 TO 11
23040 W(Z)=PEEK(VI+(40*YH)+XH)
23050 Z=Z+1:NEXTXH:NEXTYH
23060 Z=0
23070 FOR YH=20 TO 0 STEP-1
23080 FOR XH=0 TO 11
23090 POKE(VI+(40*YH)+XH),W(Z)
23100 Z=Z+1:NEXTXH:NEXTYH
```

```
23110 GOTO 20300
23999 :
24000 REM *** DEFINIEREN ***
24001 :
24010 Z=0
24020 FOR YH=0 TO 20
24030 FOR XH=0 TO 11 STEP 4
24040 CW=0:Q=0
24050 FOR T=6 TO 0 STEP-2
24055 H1=PEEK(VI+(40*YH)+XH+Q)
24060 IF H1<>174 THEN CW=CW+(H1-176)*2↑T

24070 Q=Q+1:NEXTT
24080 POKE 49152+(BL*64)+Z,CW:Z=Z+1
24090 NEXTXH:NEXTYH:GOTO20300
24999 :
25000 REM *** EDITIEREN ***
25001 :
25010 K1=49152:L1=64:YH=0:XH=0:FOR Z=0 T
O 62
25020 Q=0:FOR T=6 TO 0 STEP-2
25025 IF  PEEK(K1+(BL*L1)+Z)=0 THEN T=0:
GOTO 25050
25027 H1=(PEEK(K1+(BL*L1)+Z)AND(3*2↑T))
25030 IF H1<>0 THEN POKE (VI+(40*YH)+XH+
Q),((H1/2↑T)+176)
25050 Q=Q+1:NEXTT
25055 XH=XH+4:IFXH=12THEN XH=0:YH=YH+1
25060 NEXT Z
25080 GOTO 20300
25999 :
26000 REM *** SPRITE AUSTAUSCHEN ***
26001 :
26020 PRINT"▒▒▒▒▒▒▒▒▒▒"TAB(29)"▒SWAP:"
26025 PRINT TAB(29)"VON ";:INPUT "";B1
26030 PRINT TAB(29)"NACH";:INPUT "";B2
26070 FOR Z=0 TO 63
26080 W(Z)=PEEK(49152+(B1*64)+Z)
26090 NEXT Z
26095 FOR Z=0 TO 63
26096 H1=PEEK(49152+(B2*64)+Z)
26097 POKE 49152+(B1*64)+Z,H1
26098 NEXT Z
26099 FOR Z=0 TO 63
26100 POKE 49152+(B2*64)+Z,W(Z)
26110 NEXT Z
26112 PRINT"▒▒▒▒▒▒▒▒▒▒"TAB(29)"        "
26113 PRINT TAB(29)"        "
26114 PRINT TAB(29)"        "
26120 GOTO20300
26999 :
27000 REM *** DATA-EINGABE ***
27001 :
27010 Z=0
27020 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(25)"▒DAT
A IN:"
27040 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(24)"
      " :IF Z<0 THEN Z=0
27041 IF Z>62 THEN Z=62
27042 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(24)Z"WE
RT ?"; PEEK(49152 +(BL*64)+Z)
27043 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(24)Z"WE
RT ";
27050 INPUT "";H$
27055 IF H$="-"THEN Z=Z-1:GOTO 27020
27056 IF H$="*"THEN Z=62:GOTO 27100
27060 H1=VAL(H$):IF H1<0 THEN H1=0
27097 POKE 49152+(BL*64)+Z,H1
27100 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(24)"
      "
27110 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(24)"
      "
27120 Z=Z+1:IF Z>62 THEN 20300
27130 GOTO 27020
27999 :
28000 REM *** ZEICHENSATZ AENDERN ***
28010 :
28011 POKE 53280,1:POKE 53281,1:PRINT"▒▒
  ZEICHENSATZ AENDERN";CHR$(8);
28012 PRINTTAB(30)"▒↑ MENUE";
28014 VI=52305:FA=55377
28015 CM=7:XM=0:XP=0:YP=0:YM=0:CS=0:CG=1

28016 TA=0:PRINTCHR$(142)
28018 FORI =0 TO 15
28019 PRINT
28020 PRINT" ▒▒................▒  ▒";I*1
5;
28030 NEXTI
28040 PRINT:PRINTTAB(19)I*15;
28050 :
28100 REM *** ZEICHENSATZ ANZEIGEN ***
28101 :
28110 Z=0:FORU=0TO16:P=U*40
28120 FORI=52328+P TO 52342+P:POKEI,Z:Z=
Z+1:POKE I+3072,14:NEXT I,U
28200 PRINT"▒▒▒▒1"TAB(17)"2"
28205 PRINT"▒▒▒▒▒▒▒▒3"TAB(17)"4"
28300 PRINT"▒▒▒▒▒▒▒"
28310 PRINT" 1 SETZE   PUNKT"
28320 PRINT" ← LOESCHE PUNKT"
28330 PRINT" C CLEAR                 T ZEICH
ENSATZ G/K"
28340 PRINT" X X-SPIEGELN            E EDITI
EREN"
28350 PRINT" Y Y-SPIEGELN            = DEFIN
IEREN"
28360 PRINT" R ROTIEREN   "
28370 PRINT" I INVERTIEREN";
28380 POKE FA,0
28400 GET E$:IF E$="" THEN 28400
28410 IF E$="▒"AND XP<15 THEN XP=XP+1:GO
TO29000
28415 IF E$="▒"AND XP>0  THEN XP=XP-1:GO
TO29000
28420 IF E$="▒"AND YP<15 THEN YP=YP+1:GO
TO29000
28425 IF E$="▒"AND YP>0  THEN YP=YP-1:GO
TO29000
28430 IF E$="1" THEN POKE (FA+(40*YM)+XM
),6:CM=6:GOTO28400
28435 IF E$="←" THEN POKE (FA+(40*YM)+XM
),7:CM=7:GOTO28400
28440 IF E$="X" THEN GOSUB 30000:GOTO310
00
```

```
28445 IF E$="Y" THEN GOSUB 30000:GOTO320
00
28450 IF E$="R" THEN GOSUB 30000:GOTO330
00
28455 IF E$="I" THEN GOSUB 30000:GOTO340
00
28460 IF E$="E" THEN GOSUB 30000:GOTO350
00
28465 IF E$="C" THEN 28000
28470 IF E$="T" THEN 36000
28475 IF E$="=" THEN GOSUB 30000:GOTO370
00
28480 IF E$="↑" THEN 10130
28995 GOTO 28400
29000 REM *** CURSOR ZEIGEN ***
29010 POKE(FA+(40*YM)+XM),CM
29020 YM=YP:XM=XP:CM=PEEK(FA+(40*YM)+XM)

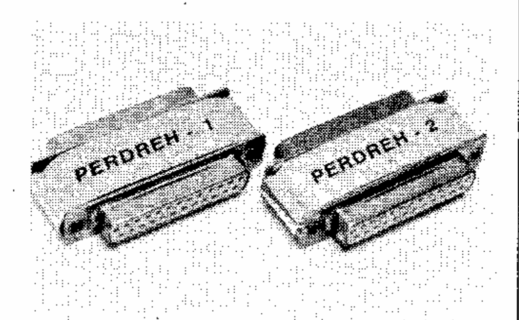
29030 POKE(FA+(40*YP)+XP),0
29040 GOTO 28400
30000 REM UNTERPROGRAMM ABFRAGE
30010 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"TA
B(22)"FENSTER NR. ";
30020 GET E$:IF E$=""THEN 30020
30025 IF E$<>"1" AND E$<>"2" AND E$<>"3"
 AND E$<>"4" AND E$<>"A" THEN 30020
30030 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"TA
B(22)"                         ";
30031 POKE (FA+(40*YP)+XP),CM
30032 IF E$="1"THEN F1=FA:L1=7
30033 IF E$="2"THEN F1=FA+8:L1=7
30034 IF E$="3"THEN F1=FA+(8*40):L1=7
30035 IF E$="4"THEN F1=FA+8+(8*40):L1=7
30036 IF E$="A"THEN F1=FA:L1=15
30040 RETURN
30999 :
31000 REM *** X-SPIEGELUNG ***
31001 :
31010 Z=0
31020 FOR YH=0 TO L1
31030 FOR XH=0 TO L1
31040 W(Z)=(PEEK(F1+(40*YH)+XH)AND15)
31050 Z=Z+1:NEXTXH:NEXTYH
31060 Z=0
31070 FOR YH=0 TO L1
31080 FOR XH=L1 TO 0 STEP-1
31090 POKE(F1+(40*YH)+XH),W(Z)
31100 Z=Z+1:NEXTXH:NEXTYH
31110 CM=(PEEK(FA+(40*YP)+XP)AND15):GOTO
 28400
31999 :
32000 REM *** Y-SPIEGELUNG ***
32001 :
32010 Z=0
32020 FOR YH=0 TO L1
32030 FOR XH=0 TO L1
32040 W(Z)=(PEEK(F1+(40*YH)+XH)AND15)
32050 Z=Z+1:NEXTXH:NEXTYH
32060 Z=0
32070 FOR YH=L1 TO 0 STEP-1
32080 FOR XH=0 TO L1
32090 POKE(F1+(40*YH)+XH),W(Z)
32100 Z=Z+1:NEXTXH:NEXTYH
32110 CM=(PEEK(FA+(40*YP)+XP)AND15):GOTO
 28400
32999 :
33000 REM *** ROTIEREN ***
33001 :
33010 Z=0
33020 FOR YH=0 TO L1
33030 FOR XH=0 TO L1
33040 W(Z)=(PEEK(F1+(40*YH)+XH)AND15)
33050 Z=Z+1:NEXTXH:NEXTYH
33060 Z=0
33070 FOR XH=L1TO 0 STEP-1
33080 FOR YH=0 TO L1
33090 POKE(F1+(40*YH)+XH),W(Z)
33100 Z=Z+1:NEXTYH:NEXTXH
33110 CM=(PEEK(FA+(40*YP)+XP)AND15):GOTO
 28400
33999 :
34000 REM *** INVERTIEREN ***
34001 :
34010 FOR YH=0 TO L1
34020 FOR XH=0 TO L1
34030 IF(PEEK(F1+(40*YH)+XH)AND15)=7 THE
N POKE (F1+(40*YH)+XH),6:GOTO 34050
34040 IF(PEEK(F1+(40*YH)+XH)AND15)=6 THE
N POKE (F1+(40*YH)+XH),7
34050 NEXTXH:NEXTYH:CM=(PEEK(FA+(40*YP)+
XP)AND15):GOTO 28400
34999 :
35000 REM *** EDITIEREN ***
35001 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"TAB
(21)"■ZEICHEN NR. ";
35002 INPUT "";ZE
35004 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"TAB
(21)"                ";
35010 K1=57344+TA:LI=8:YH=0:XH=0:FOR Z=0
 TO 7
35020 Q=0:FOR T=7 TO 0 STEP-1
35025 IF  USR(K1+(ZE*LI)+Z)=0 THEN T=0:G
OTO 35050
35030 IF (USR(K1+(ZE*LI)+Z)AND2↑T)=2↑T T
HEN POKE (F1+(40*YH)+XH+Q),6
35050 Q=Q+1:NEXTT
35055 XH=XH+8:IFXH=8 THEN XH=0:YH=YH+1
35060 NEXT Z
35080 GOTO 28400
35999 :
36000 REM ** ZEICHENSATZ UMSCHALTEN **
36001 :
36010 IF TA=0 THEN TA=2048:PRINTCHR$(14)
;:GOTO 28400
36020 IF TA=2048 THEN TA=0:PRINTCHR$(142
);:GOTO28400
36999 :
37000 REM *** DEFINIEREN ***
37001 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"TAB
(21)"■ZEICHEN NR. ";
37002 INPUT "";ZE
37004 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQ"TAB
(21)"                ";
37010 Z=0:POKE (F1+(40*YP)+XP),CM
```

```
37020 FOR YH=0 TO 7
37030 FOR XH=0 TO 7 STEP 8
37040 CW=0:Q=0
37050 FOR T=7 TO 0 STEP-1
37060 IF(PEEK(F1+(40*YH)+XH+Q)AND15)=6 T
HEN CW=CW+2↑T
37070 Q=Q+1:NEXTT
37080 POKE 57344+TA+(ZE*8)+Z,CW:Z=Z+1
37090 NEXTXH:NEXTYH:GOTO28400
38000 REM ** SAVE SPRITES+ZEICHSATZ **
38010 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
10)"R FILENAME ";
38020 INPUT "";FI$
38030 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
10)"   WARTE 4 MINUTEN          "
38040 F2$="@:"+FI$+",S,W":OPEN 1,8,4,F2$

38050 FORI=49152 TO 52224
38060 E=PEEK(I):PRINT#1,E
38070 NEXTI
38080 FORI=57344 TO 61440
38090 E=USR(I):PRINT#1,E
38100 NEXTI:CLOSE1:GOTO 10130
39000 REM ** LOAD SPRITES+ZEICHSATZ **
39010 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
10)"R FILENAME ";
39020 INPUT "";FI$
39030 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQQQQ"TAB(
10)"   WARTE 4 MINUTEN          "
39040 F2$=FI$+",S,R":OPEN 1,8,4,F2$
39050 FORI=49152 TO 52224
39060 INPUT#1,E:POKE I,E
39070 NEXTI
39080 FORI=57344 TO 61440
39090 INPUT#1,E:POKE I,E
39100 NEXTI:CLOSE1:GOTO 10130
39999 :
40000 REM *** PROGRAMM ERZEUGEN ***
40001 :
40010 PRINT"□R*** PROGRAMM GENERATOR ***
"
40020 PRINT"SPRITES:    (VON,BIS) ";:INPU
T "";A1,E1
40030 PRINT"□R              WAIT 5 MIN... ":P
OKE 648,4 :PRINT
40040 A=49152+(A1*64):REM ANFANGSADR.
40042 E=49215+(E1*64):REM ENDADRESSE
40044 ZN=55010:REM ZEILENNR.
40050 IF A1>E1 THEN 40121
40060 PRINT"□55009 REM *** SPRITES "A1"B
IS"E1" ***"
40070 PRINT"ZN="ZN":E="E":A="A":GOTO 400
80":GOTO 40800
40080 PRINT"□";ZN" FORI="A"TO"E":READ DA
:POKE I,DA:NEXTI":PRINT"ZN="ZN":E="E":A=
"A;
40090 PRINT":GOTO 40100":GOTO 40800
40100 ZN=ZN+1
40115 PRINT"□";:IF A>E THEN 40121
40116 PRINTZN"DATA ";:FORO=1 TO 17:D=PEE
K(A):D$=STR$(D):S$=""
40117 FORI=1 TO 4:E$=MID$(D$,I,1):IF E$<
>" " THEN S$=S$+E$
40118 NEXTI:PRINT S$",";:A=A+1:IF A>E TH
EN O=17
40119 NEXTO:PRINT"■ ":PRINT"A="A":ZN="ZN
+1":E="E":GOTO 40115":GOTO 40800
40121 REM *** ZEICHEN-GENERATOR ***
40125 F=0:FOR I=0 TO 4095:IF USR(53248+I
)<>USR(57344+I) THEN I=4096:F=1
40126 NEXTI:IF F=0 THEN:PRINT"□";:GOTO40
245
40130 ZN=ZN+1:PRINT"□";ZN" REM *** ZEICH
ENSATZ AENDERN ***"
40140 PRINT"ZN="ZN+1":GOTO 40150":GOTO 4
0800
40150 ZE=0
40160 AD=ZE*8:IF ZE>511 THEN 40210
40170 F=0:FORI =AD TO AD+7:IF USR(53248+
I)<>USR(57344+I) THEN F=1
40180 NEXTI:IF F=0 THEN ZE=ZE+1:GOTO 401
60
40190 PRINT"□"ZN"DATA "ZE",";:FORO=0 TO
7:D=USR(AD+57344):D$=STR$(D):S$=""
40191 FORI=1 TO 4:E$=MID$(D$,I,1):IF E$<
>" " THEN S$=S$+E$
40192 NEXTI:PRINT S$",";:AD=AD+1
40193 NEXTO:PRINT"■ ":PRINT:PRINT"ZN="ZN
+1":ZE="ZE":GOTO 40200":GOTO 40800
40200 ZE=ZE+1:GOTO 40160
40210 ZN=ZN+1:PRINT"□"ZN"DATA -1":PRINT"
ZN="ZN+10":GOTO 40220":GOTO 40800
40220 PRINT"□"ZN" READ ZE":ZN=ZN+1
40230 PRINTZN" IF ZE<>-1 THEN FORI=0TO7:
READ DA:SYS32771,(53248+(ZE*8)+I),DA:NEX
T"
40240 ZN=ZN+1:PRINTZN"IF ZE<>-1 THEN "ZN
-2
40245 PRINTZN+10" RETURN"
40250 PRINT"GOTO 40260":GOTO 40800
40260 REM *** DESIGNER LOESCHEN ***
40270 FORI=10000 TO 60000
40280 IF PEEK(I)=93 AND PEEK(I+3)=93 THE
N POKE I+4,0:U=I:I=60000
40290 NEXT
40300 POKE 648,204:U=U+5:H=INT(U/256):L=
U-(H*256)
40310 PRINT"□POKE 43,"L":POKE 44,"H
40320 PRINT"QQQQ 1 POKE 56,128:REM BASIC
 AB $8000"
40330 PRINT"2 GOSUB 50000"
40340 PRINT"LIST"
40350 GOTO 40800
40800 POKE 631,19:FORI=632 TO 640:POKE I
,13:NEXT:POKE 198,9:END
40995 :
40996 :
40997 REM ]]]]
40999 END
50000 REM VIDEOCONTROLLER UMSTELLEN
50001 REM SPRITES AB      49152 = $C000
50002 REM SPRITESPOINTER  53240 = $CFF8
```

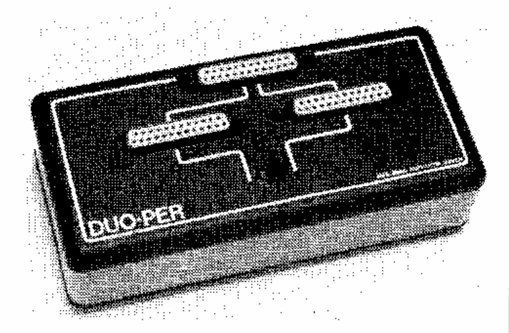
Adapter für V 24-Schnittstellen

Unter der Bezeichnung *PERDREH-1* und *PERDREH-2* sind beim Ing.-Büro Auffarth zwei Adapterstecker für V 24-Schnittstellen (25polige Sub-



min.-D-Stecker) erhältlich. Der Adapter PERDREH-1 vertauscht intern die Anschlüsse 2 und 3, läßt die übrigen 23 Anschlüsse direkt durchgehen. PERDREH-2 dreht zusätzlich noch die Anschlüsse 4 und 5. Beide Adapter können einfach zwischen Gerät und Kabel gesteckt werden. Damit ist es möglich, ohne Umlötarbeiten, mit der gleichen Kabelverdrahtung, Peripheriegeräte mit unterschiedlicher Stekkerbelegung zu betreiben. *Gesehen bei: Ing.-Büro Auffarth, Rhalandstr. 54 B, 2730 Zeven.*

V 24-Schnittstellen-umschalter



Unter der Bezeichnung *DUOPER* hat das Ing.-Büro Auffarth einen Umschalter für V 24-Schnittstellen neu in sein Produktionsprogramm aufgenommen. Das Gerät hat drei 25pol. D-Stekker, an die der Computer und zwei Peripheriegeräte angeschlossen werden. Mit dem eingebauten Schalter wird zwischen den Peripheriegeräten umgeschaltet. Es entfällt damit das lästige Umstecken der Peripheriegeräte, wenn abwechselnd eine Computerschnittstelle benutzt werden muß. *Gesehen bei: Ing.-Büro Auffarth, Rhalandstr. 54 B, 2730 Zeven.*

```
50003 REM VIDEOSCREEN AB  52224 = $CC00
50004 REM ZEICHENSATZ AB  53248 = $D000
50005 REM
50100 FORI= 32768 TO 32900 :READ DA:POKE
I,DA:NEXTI:SYS 32768
54990 REM *** MASCHINENPROGRAMM ***
55000 DATA 76,6,128,76,73,128,120,173,0,
221,41,252,141,0,221,169,52
55001 DATA 141,24,208,169,204,141,136,2,
169,97,141,17,3,169,128,141,18
55002 DATA 3,160,0,132,3,169,208,133,4,1
62,16,169,51,133,1,177,3
55003 DATA 72,169,48,133,1,104,145,3,200
,208,239,230,4,202,208,234,169
55004 DATA 55,133,1,88,96,120,165,1,72,3
2,253,174,32,235,183,169,48
55005 DATA 133,1,160,0,138,145,20,104,13
3,1,88,96,165,20,72,165,21
55006 DATA 72,32,247,183,165,1,72,169,52
,120,133,1,160,0,177,20,168
55007 DATA 104,133,1,88,104,133,21,104,1
33,20,76,162,179,255
55011 RETURN
```

Homeword – die persönliche Textverarbeitung

für VC-64

Goldmedaillen für die USA – das gilt nicht nur für die vergangenen Olympischen Spiele, sondern auch für gute Software, die immer noch zuerst aus Übersee kommt. Goldmedaillengewinner ist auch das Textverarbeitungsprogramm HOMEWORD von Sierra Online für Commodore 64 und Apple II, nach einem Vergleich des Software Buyer's Guide der „Word Processors for the Home".

HOMEWORD ist der Bestseller der Textverarbeitungsprogramme für den Commodore 64 in den USA und heimst vor allem hohes Lob für seine Benutzerfreundlichkeit, die einfache und übersichtliche Bedienung des Programms: „HOMEWORD is easier to use than a game" so „Creative Computing".

HOMEWORD arbeitet ganz modern mit Bildsymbolen bei den einzelnen Menüs und führt so auch den Anfänger einfach und sicher durchs Programm. Eines der größten Komplimente lautet denn auch: HOMEWORD verbannt die Furcht vor dem Computer (Creative Computing).

So auch „Classroom Computer Learning": „HOMEWORD bedeutet einen innovativen Durchbruch von Textverarbeitung in der Schule und zu Hause. Es ist von außerordentlichem Wert für alle, die ein anspruchsvolles aber leicht zu erlernendes Programm wollen."

HOMEWORD bietet trotz seiner einfachen Bedienung natürlich alles, was der Benutzer von einer guten, modernen Textverarbeitung erwartet: Komfortable Korrektur- und Suchvorgänge, professionelle Umbruch- und Formatierungsmöglichkeiten, Seitennumerierung – so daß lt. der Zeitschrift „Personal Software" „dieser Wordprocessor dem Anfänger Einfachheit bietet ohne den Fachmann zu frustrieren".

Diese Eigenschaften mögen auch IBM bewogen haben, HOMEWORD als offiziellen Wordprocessor für den PC junior auszuwählen.

Bisher liegen „nur" begeisterte Stimmen der amerikanischen Fachpresse vor, aber Ähnliches wird wohl auch bald bei uns zu hören sein: Der Langenscheidt-Verlag hat die Lizenz für die deutschsprachige Version dieses Erfolgsprogramms erworben und wird die Ausgabe für den deutschen Markt Anfang des Jahres 1985 anbieten.

Da auch bei uns der Trend für Schul- und Homecomputer zu höchstmöglicher Benutzerfreundlichkeit geht, dürfte auch hier mit einem Siegeszug des „problemlosen" HOMEWORD zu rechnen sein.

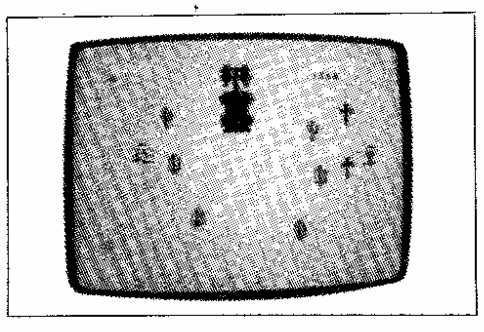
Ein superschnelles Spiel für den VC-64.

```
0 REM    COPYRIGHT  J.BAAS  9.9.1984
1 POKE56,144:CLR:PRINT"♥"
2 DATA120,169,51,133,1,169,0,133,95,133,
90,133,88,169,208,133,96,169,240
3 DATA133,89,169,224,133,91,32,191,163,1
69,55,133,1,88,96
4 FORI=832TO832+33
5 READA:POKEI,A:NEXT
6 SYS832:POKE850,160:SYS832
7 POKE53272,8:POKE56576,PEEK(56576)AND25
2:POKE648,192
8 PRINTCHR$(8)
9 :
10 POKE53280,1:POKE53281,1:PRINTCHR$(14)

20 POKE53248+21,0:PRINT"♥▮LEASE WAIT A
MOMENT":GOSUB55000:GOSUB55500:DD=0
30 PRINT"♥":PRINTCHR$(142):POKE58280,0:P
OKE53281,0:L1=0:L2=0
31 PRINT"♥▮▮▮▮▮▮▮▮"SPC(6)"(I)NSTRUCTIONS
":PRINTTAB(6)"(O)PTIONS":PRINTTAB(8)"O
R"
32 PRINTTAB(6)"(G)AME"
33 GETQQ$:IFQQ$="G"THEN40
34 IFQQ$="I"THEN5000
35 IFQQ$="O"THEN6000
37 GOTO33
40 PRINT"♥▮▮▮▮▮▮  ┌──┐"
50 PRINT" │   └─┐ ╭──┐ ╭───╮ ┌───╮ ┌──┐
 ╭──┐"
60 PRINT" │   ┌─┘ │ ╭─┘ │╭─╮│ │ ┌╮ │ │ │
 │ ╭─┘"
70 PRINT" │   │   │ │   ││ ││ │ ││ │ │ │
 │ │"
80 PRINT" │   └┐  │ │   │╰─╯│ │ ││ │ │ │
 │ └─┐"
90 PRINT" └────┘   └─┘   ╰───╯ └─┘└─┘ └──┘
 └────┘"
100 SI=54272:S1=SI:S2=54279:S3=54286:FL=
54293:FH=54294:RS=54295:PL=54296
110 POKES1+4,0:POKES2+4,0:POKES3+4,0
111 A=0:D=1:S=13:R=10:H=500
112 POKES1+5,16*A+D:POKES1+6,16*S+R
113 POKES2+5,16*A+D:POKES2+6,16*S+R
114 POKES3+5,16*A+D:POKES3+6,16*S+R
115 POKERS,0:POKEPL,15
116 POKES1,37:POKES1+1,15
117 POKES2,154:POKES2+1,21
118 POKES3,177:POKES3+1,25
119 POKES1+4,33:FORI=0TO400:NEXT:POKES2+
4,33:FORI=0TO400:NEXT:POKES3+4,33
120 FORI=0TOH:NEXT:POKES1+4,32:POKES2+4,
32:POKES3+4,32
130 POKEFH,4:POKEFL,226:POKEW,33+1:FORI=
1TO2000:NEXT:POKEW,0
135 PRINT"▮▮▮"SPC(20)"▮PRESENTS:":FORI=1
TO3000:NEXT:PRINT"♥"
140 FORI=0TO15:POKEV+I,0:NEXT:POKEV+21,63
```

Das Programm ist vollständig in Maschinensprache geschrieben. Mit 255 verschiedenen Geschwindigkeitsstufen!!!

Zwei Cowboys stehen sich auf einer vielbefahrenen Postkutschenstraße gegenüber. Da jeder von ihnen zuerst auf die andere Seite will und keiner nachgibt, kommt es zum Duell.



Jede Spielfigur hat einen Revolver mit sechs Schuß Munition. Ist der Revolver leergeschossen, dauert es eine gewisse Zeit, bis er wieder geladen ist. Ist eine Spielfigur dreimal getroffen worden, ist sein Gegenüber der Sieger und darf die Straße überqueren.

High Noon ist ein sehr schnelles Spiel, da es vollständig in Maschinensprache geschrieben wurde. Nur der Vorspann und die Auswertung sind in Basic geschrieben. Die Spiel- und Schußgeschwindigkeit können frei gewählt werden. Zu beachten ist, daß ein Schuß mit hoher Geschwindigkeit auch einen Kaktus durchschlägt, der ansonsten Schutz gewährt. Die höchste, erreichbare Spielgeschwindigkeit ist >255!!!<, die niedrigste ist >0<. Zur Grafik ist zu sagen, daß sie durch neue Grafikzeichen, eine neue Schrift und Multicolor-Sprites sehr schön ist. Das Spiel ist für zwei Personen gedacht und wird mit Joystick gesteuert.

```
150 POKE53280,7:POKE53281,7:PRINT"   SHO
OTS"TAB(30)"SHOOTS"
160 PRINT"    KKKKKK"TAB(30)"KKKKKK"
170 A$="AB C D E F":B$="A G H
I J":K$="M N O P Q R"
175 KW$="               "
180 PRINT""TAB(9)A$;TAB(29)A$
190 PRINT""TAB(10)B$;TAB(30)B$
200 PRINT""TAB(13)B$;TAB(27)B$
210 POKEV,25:POKEV+1,229:POKEV+16,34:IFD
D<>0THENGOTO340
220 FORI=229TO80STEP-1:IFI/2=INT(I/2)THE
N240
230 POKE50168,16:POKEV+1,I:GOTO250
240 POKE50168,17:POKEV+1,I
250 SYS36864:FORO=1TO30:NEXTO:NEXTI
260 POKE50168,18:PRINT"":FORI=1TO9:S
YS36903
270 PRINTTAB(15+I);A$(I);:FORO=1TO200:NE
XTO:NEXTI
280 FORI=80TO112:IFI/2=INT(I/2)THEN300
290 POKE50168,16:POKEV+1,I:GOTO310
300 POKE50168,17:POKEV+1,I
310 SYS36864:FORO=1TO50:NEXTO:NEXTI
320 POKE50168,18:PRINT"":FORI=1TO8:SY
S36903
330 PRINTTAB(15+I);A$(I+9);:FORO=1TO200:
NEXTO:NEXTI
340 POKEV+21,63:POKEV+2,64:POKEV+16,34:P
OKEV+3,229:POKEV+30,0:POKEV+31,0
350 POKEV+4,155:POKEV+5,149:POKEV+6,155:
POKEV+7,149+42
351 POKE40953,0:POKE37536,76
355 POKE40959,6:POKE40958,6:POKE40957,24
0:POKE40956,240:POKE40955,0:POKE40954,0
357 PRINT"":PRINTTAB(16)"
":PRINT""TAB(16)"                "
360 POKE56322,224:SYS36944:POKE56322,255
370 DD=10
380 POKE56322,255:FORI=8TO11:POKEV+I,0:N
EXT
390 T=PEEK(40953):IFT=1THEN:L1=L1+1:GOTO
410
400 L2=L2+1
410 PRINT"":IFL1<>0THENFORI=1TOL1:PRINT
TAB(2)""K$:NEXT
415 PRINT"":IFL2<>0THENFORI=1TOL2:PRINT
TAB(34)""K$:NEXT
416 POKES1+4,0:POKES2+4,0:POKES3+4,0
417 A=0:D=1:S=13:R=10:H=500
418 POKES1+5,16*A+D:POKES1+6,16*S+R
419 POKES2+5,16*A+D:POKES2+6,16*S+R
420 POKES3+5,16*A+D:POKES3+6,16*S+R
421 POKERS,0:POKEPL,15
422 POKES1,37:POKES1+1,15
423 POKES2,154:POKES2+1,21
424 POKES3,177:POKES3+1,25
425 POKES1+4,17:FORI=0TO400:NEXT:POKES2+
4,17:FORI=0TO400:NEXT:POKES3+4,17
426 FORI=0TOH:NEXT:POKES1+4,16:POKES2+4,
16:POKES3+4,16
429 IFL1=3THENW=2:GOSUB500:GOTO450
430 IFL2=3THENW=1:GOSUB700:GOTO450
440 FORI=1TO2000:NEXT:PRINT"":FORI=1TO3
:PRINTTAB(2)""KW$:NEXT
441 PRINT"    KKKKKK"TAB(30)"KKKKKK"
442 PRINT"":FORI=1TO3:PRINTTAB(34)""
"KW$:NEXT
445 POKEV+1,90:GOTO340
450 PRINT"":POKEV+21,0
460 PRINT"COWBOY"W"IST DER SIEGER":PR
INT"WAGEN SIE NOCH EIN DUELL ?"
470 GETQ$:IFQ$="J"THENL1=0:L2=0:GOTO30
480 IF Q$="N"THENPRINT"LOESCHEN SIE D
AS SPIEL MIT:":PRINT"SYS 64738":PRINT"
":END
490 GOTO470
500 POKEV+21,2:P=PEEK(V+2)
510 FORI=PTO0STEP-1:POKE(V+2),I:SYS36864
:IFI/2=INT(I/2)THENPOKE50169,16:GOTO530
520 POKE50169,17
530 NEXT:POKEV+16,0
540 FORI=255TO0STEP-1:POKE(V+2),I:IFI/2=
INT(I/2)THENPOKE50169,16:GOTO560
550 POKE50169,17
560 SYS36864:NEXT:POKEV+21,0:RETURN
700 POKEV+21,1:P=PEEK(V)
710 FORI=PTO255:POKE(V),I:IFI/2=INT(I/2)
THENPOKE50168,16:GOTO730
720 POKE50168,17
730 SYS36864:NEXT:POKEV+16,1
740 FORI=0TO64:POKE(V),I:IFI/2=INT(I/2)T
HENPOKE50168,16:GOTO760
750 POKE50168,17
760 SYS36864:NEXT:POKEV+21,0:RETURN
4997 :
4998 REM  A N L E I T U N G
4999 :
5000 PRINT""SPC(11)"HIGH NOON       £
J.BAAS"
5010 PRINT"AN EINER STARK BEFAHRENEN P
OSTKUTSCHEN-"
5020 PRINT"DURCHGANGSSTRASSE TREFFEN SIC
H ZWEI"
5030 PRINT"COWBOYS.JEDER COWBOY WILL DIE
 STRASSE"
5040 PRINT"ZUERST UEBERQUEREN.ES KOMMT Z
UM DUELL..."
5050 PRINT"JEDER SPIELER STEUERT EINEN
COWBOY."
5060 PRINT"JEDER COWBOY HAT DREI LEBEN,B
EVOR ER"
5070 PRINT"ENTGUELTIG VERLOREN HAT."
5071 PRINT"GESCHOSSEN WIRD MIT SECHS-SC
HUESSIGEN"
5072 PRINT"REVOLVERN.WENN DIE TROMMEL LE
ER IST,"
5073 PRINT"BRAUCHT DER COWBOY EINIGE ZEI
T,BIS ER   WIEDER GELADEN HAT."
5080 PRINT""SPC(24)"VIEL GLUECK !!!"
5090 POKE198,0:WAIT198,1:GOTO30
5997 :
5998 REM  O P T I O N E N
5999 :
6000 PRINT"  GAME OPTIONS :"
6010 PRINT"1......GESCHWINDIGKEIT SCHU
SS (NORM 05)"
6020 PRINT"2......GESCHWINDIGKEIT SPIE
L (NORM.240)"
```

```
6025 PRINT"■■3......MENUE"
6030 PRINTTAB(23)"■■■BITTE WAEHLEN SIE"
6040 GETA$:IFA$="1"THENPP=37309:GOTO6100
6050 IFA$="2"THENPP=37525:GOTO6100
6060 IFA$="3"THEN30
6070 GOTO6040
6100 PRINT:PRINT:INPUT"GESCHWINDIGKEIT (
0-255)";G:IFG<0ORG>255THEN6100
6110 POKEPP,G:GOTO6000
49999 GOTO49999
54997 :
54998 REM  G R A F I K + S C H R I F T
54999 :
55000 A=53
55010 FORI=1TOA:READE:E=E+1
55020 C=(9*4096+(8*E))+20472
55030 FORN=0TO7:READQ:POKEC+N,Q:NEXTN
55040 NEXTI:RETURN
55100 DATA1,60,36,126,98,98,98,98,0
55101 DATA2,124,36,62,50,50,50,126,0
55102 DATA3,126,66,64,96,96,98,126,0
55103 DATA4,126,34,34,50,50,50,126,0
55104 DATA5,126,64,64,120,96,96,126,0
55105 DATA6,126,64,64,120,96,96,96,0
55106 DATA7,126,66,64,110,98,98,126,0
55107 DATA8,66,66,66,126,98,98,98,0
55108 DATA9,16,16,16,24,24,24,24,0
55109 DATA10,2,2,2,6,6,70,60,0
55110 DATA11,66,68,72,126,98,98,98,0
55111 DATA12,64,64,64,96,96,96,126,0
55112 DATA13,102,90,66,98,98,98,98,0
55113 DATA14,114,74,74,106,106,106,102,0
55114 DATA15,126,66,66,98,98,98,126,0
55115 DATA16,126,66,66,126,96,96,96,0
55116 DATA18,126,66,66,126,100,98,98,0
55117 DATA19,126,66,64,126,6,70,126,0
55118 DATA20,124,16,16,24,24,24,24,0
55119 DATA21,66,66,66,98,98,98,126,0
55120 DATA23,98,98,98,98,66,90,102,0
55121 DATA24,66,36,24,124,98,98,98,0
55122 DATA25,66,66,66,60,24,24,24,0
55123 DATA48,254,130,134,210,226,194,254
,0
55124 DATA49,112,16,16,48,48,48,120,0
55125 DATA50,254,130,2,254,192,192,254,0

55126 DATA51,126,2,2,62,6,6,254,0
55127 DATA52,128,128,132,132,254,12,12,0

55128 DATA53,126,64,64,126,6,70,254,0
55129 DATA54,254,128,128,254,194,194,254
,0
55130 DATA55,254,130,4,8,24,24,24,0
55131 DATA56,120,68,68,254,194,194,254,0

55132 DATA57,254,130,130,254,6,6,254,0
55133 DATA0,33,136,84,32,136,84,34,137
55134 DATA28,60,66,157,161,161,157,66,60

55135 DATA129,0,1,3,3,51,51,51,51
55136 DATA130,51,51,51,63,31,7,3,3
55137 DATA131,3,3,3,3,3,3,3,0
55138 DATA132,0,128,192,192,192,196,206,
206
55139 DATA133,206,206,206,206,206,254,25
4,248
55140 DATA134,192,192,192,192,192,192,19
2,192
55141 DATA135,51,51,51,51,51,51,51,51
55142 DATA136,51,51,63,31,3,3,3,0
55143 DATA137,206,206,206,206,206,206,20
6,206
55144 DATA138,206,206,254,248,224,192,19
2,0
55145 DATA139,0,8,28,28,28,28,28,62
55146 DATA140,0,0,0,12,0,0,0,0
55147 DATA141,0,1,3,3,1,1,97,49
55148 DATA142,63,63,49,97,1,1,1,1
55149 DATA143,1,1,1,1,1,1,1,0
55150 DATA144,0,128,192,192,128,128,134,
140
55151 DATA145,252,252,140,134,128,128,12
8,128
55152 DATA146,128,128,128,128,128,128,12
8,128
55153 :
55154 REM  S P R I T E S
55155 :
55500 V=53248:POKEV+32,8:POKEV+33,8:POKE
V+28,3
55510 POKEV+39,14:POKEV+40,14:POKEV+37,0
:POKEV+38,1:POKEV+41,0:POKEV+42,0
55511 POKEV+43,1:POKEV+29,12:POKEV+23,12
55515 POKEV,0:POKEV+1,0:POKEV+2,0:POKEV+
3,0:POKE50173,22:POKEV+44,1
55520 POKE50168,16:POKE50169,16:POKE5017
0,20:POKE50171,21:POKE50172,22
55530 FORI=0TO62:READX:POKE50176+I,X:NEX
T
55540 FORI=0TO62:READX:POKE50239+I,X:NEX
T
55550 FORI=0TO62:READX:POKE50302+I,X:NEX
T
55560 FORI=0TO65:READX:POKE50365+I,X:NEX
T
55570 DIMA$(17):FORI=1TO17:READA$(I):NEX
T
55580 FORI=0TO62:READX:POKE50432+I,X:NEX
T
55590 FORI=0TO62:READX:POKE50496+I,X:NEX
T
55600 FORI=1TO24:POKE50559+I,0:NEXT:POKE
50559+25,60
55610 FORI=1TO38:POKE50559+25+I,0:NEXT
55617 :
55618 REM    S O U N D
55619 :
55620 S=0:FORI=36864TO36943:READX:POKEI,
X:S=S+X:NEXT:IFS<>9122THENSTOP
55627 :
55628 REM    H A U P T P R O G R A M M
55629 :
55630 S=0:FORI=36944TO37550:READX:POKEI,
X:S=S+X:NEXT:IFS<>77402THENSTOP
55637 :
55638 REM   I N T E R R U P T P R .
55639 :
55640 S=0:FORI=33024TO33120:READX:POKEI,
X:S=S+X:NEXT:IFS<>12171THENSTOP
55650 RETURN
```

```
55997 :
55998 REM S P R I T E S
55999 :
56001 DATA0,20,0,0,20,0,1,85,64,0,60,0,0
,60,0,42,170,168,170,170,170,130,170
56002 DATA130,130,170,130,162,170,138,61
,85,124,6,170,144,6,130,144,6,130,144
56003 DATA2,130,128,2,129,64,2,129,64,1,
65,84,1,65,84,21,64,0,21,64,0,0,0,20
56004 DATA0,0,20,0,1,85,64,0,60,0,0,60,0
,42,170,168,170,170,170,130,170,130
56005 DATA130,170,130,162,170,138,61,85,
124,6,170,144,6,130,144,6,130,144,2
56006 DATA130,128,1,66,128,1,66,128,21,6
5,128,21,65,128,0,1,84,0,1,84,0,5,0,0
56007 DATA5,0,0,85,80,0,15,0,0,15,0,0,42
,160,0,42,160,0,42,160,21,42,170,176
56008 DATA42,160,16,21,80,0,37,160,0,9,1
28,0,9,128,0,9,128,0,10,128,0,10,128
56009 DATA0,5,64,0,5,64,0,5,84,0,5,84,0,
0,0,0,80,0,0,80,0,5,85,0,0,240,0,0,240
56010 DATA0,10,168,0,10,168,84,10,168,14
,170,168,12,10,168,0,5,84,0,10,88,0
56011 DATA2,96,0,2,96,0,2,96,0,2,160,0,2
,160,0,1,80,0,1,80,0,84,80,0,84,80
56100 DATAH,I,G,H," ",N,O,O,N,£," ",J,".
",B,A,A,S
56101 DATA1,128,96,3,192,240,1,128,96,1,
128,96,3,255,240,7,237,248,3,204,240
56102 DATA3,204,240,7,237,248,7,237,248,
3,204,240,1,140,96,1,12,32,1,12,32
56103 DATA0,12,0,0,12,0,1,255,224,5,255,
232,5,255,232,15,255,252,15,255,252
56104 DATA5,255,232,5,255,232,1,255,224,
1,255,224,1,255,224,1,255,224,1,255,224
56105 DATA5,255,232,5,255,232,15,255,252
,15,255,252,5,255,232,5,255,232
56106 DATA1,255,224,3,255,240,3,255,240,
,,,,,,,,,,,,,,,
56107 :
56108 REM S O U N D
56109 :
56110 DATA169,0,141,24,212,141,4,212,169
,10,141,24,212,169,2,141,5,212,169
56111 DATA3,141,6,212,169,40,141,1,212,1
69,200,141,0,212,169,129,141,4,212
56112 DATA96,169,0,141,11,212,169,0,141,
24,212,169,15,141,24,212,169,9,141
56113 DATA12,212,169,0,141,13,212,169,17
,141,8,212,169,15,141,7,212,169,129
56114 DATA141,11,212,96
56115 :
56116 REM M C - P R O G R A M M
56117 :
56118 DATA120,169,0,141,20,3,169,129,141
,21,3,88,32,163,146,201,15,208,3,76
56119 DATA201,144,32,0,144,173,248,195,2
01,16,240,8,169,16,141,248,195,76,125
56120 DATA144,169,17,141,248,195,173,0,2
20,41,1,208,12,173,1,208,201,71,240
56121 DATA5,233,1,141,1,208,173,0,220,41
,2,208,12,173,1,208,201,229,240,5,105
56122 DATA1,141,1,208,173,0,220,41,4,208
,12,173,0,208,201,25,240,5,233,1,141
56123 DATA0,208,173,0,220,41,8,208,12,17
3,0,208,201,64,240,5,105,1,141,0,208
56124 DATA32,169,146,201,15,208,3,76,54,
145,32,0,144,173,249,195,201,16,240
56125 DATA8,169,16,141,249,195,76,234,14
4,169,17,141,249,195,173,1,220,41,1
56126 DATA208,12,173,3,208,201,71,240,5,
233,1,141,3,208,173,1,220,41,2,208
56127 DATA12,173,3,208,201,229,240,5,105
,1,141,3,208,173,1,220,41,4,208,12
56128 DATA173,2,208,201,28,240,5,233,1,1
41,2,208,173,1,220,41,8,208,12,173
56129 DATA2,208,201,64,240,5,105,1,141,2
,208,160,3,206,5,208,206,7,208,234
56130 DATA234,234,136,208,244,173,0,220,
41,16,208,53,162,18,142,248,195,174
56131 DATA251,159,224,0,208,41,174,255,1
59,224,0,240,34,202,142,255,159,162
56132 DATA1,142,251,159,32,39,144,174,25
5,159,169,32,157,42,192,173,0,208,105
56133 DATA24,141,8,208,173,1,208,141,9,2
08,173,1,220,41,16,208,53,162,19,142
56134 DATA249,195,174,250,159,224,0,208,
41,174,254,159,224,0,240,34,202,142
56135 DATA254,159,162,1,142,250,159,32,3
9,144,174,254,159,169,32,157,70,192
56136 DATA173,2,208,233,1,141,10,208,173
,3,208,141,11,208,160,11,173,251,159
56137 DATA201,0,240,30,238,8,208,208,8,1
73,16,208,9,16,141,16,208,173,8,208
56138 DATA201,80,208,10,173,16,208,41,16
,240,3,32,14,146,173,250,159,201,0
56139 DATA240,30,206,10,208,16,8,173,16,
208,41,223,141,16,208,173,10,208,201
56140 DATA0,208,10,173,16,208,41,32,208,
3,32,34,146,136,208,179,76,54,146,173
56141 DATA16,208,41,239,141,16,208,169,0
,141,8,208,141,9,208,141,251,159,96
56142 DATA173,16,208,9,32,141,16,208,169
,0,141,10,208,141,11,208,141,250,159
56143 DATA96,173,255,159,201,0,208,40,17
3,253,159,233,1,208,30,169,6,141,255
56144 DATA159,169,0,141,251,159,169,240,
141,253,159,162,0,169,139,157,42,192
56145 DATA232,224,6,208,248,76,101,146,1
41,253,159,173,254,159,201,0,208,40
56146 DATA173,252,159,233,1,208,30,169,6
,141,254,159,169,0,141,250,159,169
56147 DATA240,141,252,159,162,0,169,139,
157,70,192,232,224,6,208,248,76,148
56148 DATA146,141,252,159,162,224,160,0,
200,208,253,232,224,0,208,246,76,92
56149 DATA144,173,0,220,41,15,96,173,1,2
20,41,15,96
56160 :
56161 REM   I N T E R R U P T
56162 :
56163 DATA173,30,208,141,248,159,173,31,
208,141,247,159,173,248,159,41,1,240
56164 DATA10,169,1,141,249,159,169,96,14
1,160,146,173,248,159,41,2,240,10,169
56165 DATA2,141,249,159,169,96,141,160,1
46,173,248,159,41,16,240,3,32,14,146
56166 DATA173,248,159,41,32,240,3,32,34,
146,173,247,159,41,16,240,3,32,14,146
56167 DATA173,247,159,41,32,240,3,32,34,
146,169,0,141,30,208,141,31,208,76
56168 DATA49,234
```



Buffalo Bill – Abenteuer im Wilden Westen.

Für VC-20 ohne Erweiterung

Wir befinden uns in der Pionierzeit des Wilden Westens. Billy the Kid und seine Bande versetzen das Land in Angst und Schrecken. Eines Tages sprengen sie eine große Eisenbahnbrücke der Grand-National-Railroad. Der Zug, der die gesamten Goldvorräte der Eisenbahngesellschaft transportierte, stürzt von den Gleisen und verunglückt. Durch die Wucht der Explosion werden die Goldvorräte in alle Richtungen fortgeschleudert und liegen nun verstreut zwischen Kakteen in der Wüste. Sofort nach dem Unglück benachrichtigt der Eisenbahndirektor Buffalo Bill. Er soll retten, was noch zu retten ist! Buffalo reitet sofort los und beginnt das Gold aufzusammeln. Doch er hat nicht mit Billy the Kid gerechnet! Dieser versucht nun mit allen Mitteln, Buffalo am Aufsammeln zu hindern. Dummerweise hat Buffalo seinen Colt vergessen, also bleibt ihm nichts anderes übrig, als fortzulaufen. Doch Vorsicht vor den Kakteen und der Umzäunung.

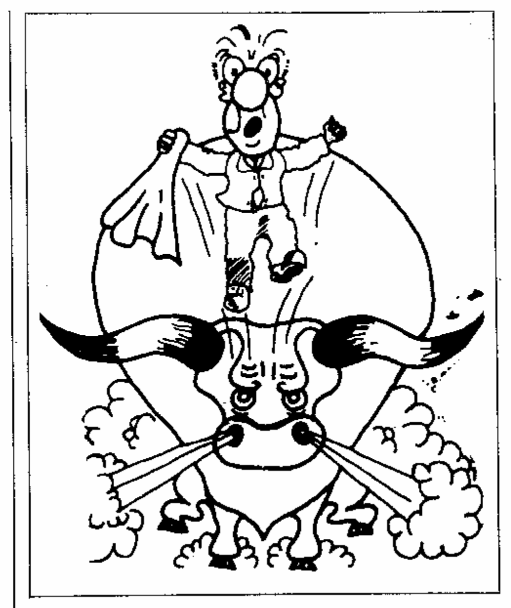
Wenn Buffalo es endlich geschafft hat, das Gold aufzusammeln und Billy the Kid zu entkommen, gerät er auf dem Weg zur Bank in den berühmt-berüchtigten Indianer-Canyon!

Pfeile aus dem Hinterhalt zischen ihm um die Ohren, und schauerliches Indianergeheul klingt aus allen Richtungen! Der erste Pfeiltreffer verwundet ihn nur, der zweite wird tödlich sein! Am Ende des Canyon erkennt Buffalo, schwer mitgenommen von den Strapazen, das Ortsschild von Buffalo-Bill-Ville.

Nun ist er endlich da; die Bank ist schon in Sichtweite, doch just in dem Augenblick, als er die Straße überqueren will, passiert ihn eine riesige Rinderherde. Da Billy the Kid immer noch hinter ihm her ist, hat er keine Zeit zu warten, bis die Herde vorbei ist!

So muß er sich also zwischen den Rindern hindurch auf die andere Seite begeben und muß aufpassen, daß er nicht auf die Hörner gerät!

Hat er die andere Straßenseite erreicht, braucht er nur noch in einen der beiden, mit Pfeilen markierten, Eingänge zu gehen. Puh, geschafft! Den Whisky im nächsten Saloon hat er sich redlich verdient!



Das Spiel füllt die 3.5 K des VC-20 bis auf wenige Bytes voll aus, darum ist beim Eintippen größte Sorgfalt angebracht, einen OUT OF MEMORY ERROR ist schnell fabriziert!

Zum Spielen benötigt wird ein Joystick, der am Controlport auf der rechten Seite des VC-20 angeschlossen wird.

```
1 REMSTEP▚GOSUB"▜▘ ▜)▄▄▛π▩"ON/ON)▔ANDINPUTAND/ON) INPUTOR-
5 GOTO20000
9 GOSUB21000:CLR
10 POKE36879,29:PRINT"▜▩":XX=2
20 C=7922:POKEC,81:S(1)=95:S(2)=105:T(1)=183:T(2)=185
25 S$="▩ "+CHR$(20)+"▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩▩ "+CHR$(20)+"▩"
26 POKE36878,1
27 FORI=38400TO38906:POKEI,2:NEXT
30 X=INT(RND(1)*21)+8120
40 Y=INT(RND(1)*20)*21+7724
50 FORL=1TO19
55 S=S+1:IFS=3THENS=1
56 POKE7701,S(S):POKE8185,S(S)+128
57 PRINTS$
58 POKE36875,T(S)
60 POKEX,30:POKEY,31:POKEX+22,32:POKEY+1,32
70 X=X-22:Y=Y-1
71 SYS4102:IFPEEK(251)<>156THENGOTO100
75 IFPEEK(C)<>81THEN170
80 NEXT
90 POKEY+1,32:POKEX+22,32:GOTO30
100 IFPEEK(251)=28THENR=1:GOTO150
110 IFPEEK(251)=148THENR=22:GOTO150
120 IFPEEK(251)=140THENR=-1:GOTO150
130 IFPEEK(251)=152THENR=-22:GOTO150
140 GOTO80
150 IFPEEK(C+R)<>32ORPEEK(C)<>81THEN170
160 POKEC,32:C=C+R:POKEC,81:POKEC+30720,0:F=F+1:IFF<100THEN80
165 GOTO5000
```

```
170 POKE36875,0:POKE36878,10:POKEC,42:POKEC+30720,4:FORI=135TO195:POKE36877,I:NE
XT
180 POKE36877,0:POKE36878,1:XX=XX-1:IFXX<=0THEN10000
190 POKEC,32:C=7922:POKEC,81:GOTO80
5000 CLR:POKE36875,0:POKE36878,10:GOSUB21000
5020 POKE36879,30
5030 S$="":FORI=1TO10:S$=S$+"  "+CHR$(20)+"":NEXT
5040 PRINT"      ______"
5050 PRINT"  __  BANK  __"
5060 PRINT"  ............. "
5070 PRINT"  ............. "
5080 PRINT"              "
5090 PRINT"     ┌┐    ┌┐    "
5095 PRINT"    ↑    ↑"
5105 A(0)=32:A(1)=32:A(2)=32:A(3)=32:A(4)=32:A(5)=32:A(6)=32:A(7)=81
5106 X=8175:POKEX,87:POKEX+30720,2
5110 FORI=1TO22:FORJ=7921TO8119STEP22:POKEJ,A(INT(RND(1)*8)):NEXT:PRINTS$:NEXT
5120 SYS4102:IFPEEK(251)=156THENR=0:GOTO5220
5130 IFPEEK(251)=152THENR=-22:GOTO5200
5140 IFPEEK(251)=28THENR=1:GOTO5200
5150 IFPEEK(251)=148THENR=22:GOTO5200
5160 IFPEEK(251)=140THENR=-2:GOTO5200
5170 GOTO5220
5200 POKEX,32
5220 FORI=7921TO8119STEP22:POKEI,A(INT(RND(1)*8)):NEXT:PRINTS$
5225 IFPEEK(X)=81THEN10000
5230 POKEX-1,32:X=X+R:POKEX,87:POKEX+30720,2
5235 IFX=7795ORX=7801THEN15000
5240 GOTO5120
10000 FORI=25TO31:POKE36879,I:FORJ=135TO195:POKE36877,J:NEXT:NEXT:POKE36877,0
10010 POKE36879,30:PRINT"SIE HABEN PECH GEHABT! DAS GOLD IST FUTSCH!"
10011 GOTO15010
15000 PRINT"       GRATULIERE !":PRINT"SIE HABEN ES GESCHAFFT"
15005 FORR=1TO3:GOSUB21000:NEXT
15010 FORI=1TO3000:NEXT
20000 POKE36879,59:PRINT"WESTERN ADVENTURE"
20010 PRINT"BRINGEN SIE DIE GOLD- VORRAETE DER EISEN-   BAHN SICHER ZUR BANK
 IN"
20015 PRINT"     ,______________,"
20020 PRINT"     |BUFFALO-BILL|       |   VILLE    |"
20030 PRINT"     `______________'"
20040 FORI=1TO4:PRINT"              ":NEXT
20050 PRINT"    - BITTE TASTE -"
20060 POKE198,0:WAIT198,1
20070 POKE36879,25
20075 QQ=1024:Q=QQ
20080 PRINT"######################";
20090 FORI=1TO21:PRINT"#                    #";:NEXT
20100 PRINT"#####################"
20110 POKE38905,0:POKE8185,35
20115 FORI=1TO10
20120 GD=INT(RND(1)*506+7680):IFPEEK(GD)<>32THEN20120
20125 POKEGD,81:POKEGD+30720,7
20130 KA=INT(RND(1)*506)+7680:IFPEEK(KA)<>32THEN20130
20135 POKEKA,65:POKEKA+30720,5
20140 NEXT:GD=0
20150 X=INT(RND(1)*506+7680):IFPEEK(X)<>32THEN20150
20160 POKEX,87
20170 Y=INT(RND(1)*506+7680):IFPEEK(Y)<>32THEN20170
20180 POKEY,88
20190 W(1)=-22:W(2)=-1:W(3)=-22:W(4)=1:W(5)=22:W(6)=-1
20200 POKE36878,10:FORI=1TO20:POKE36876,201:POKE36876,207:NEXT:POKE36876,0
20210 SYS4102:IFPEEK(251)=156THEN20500
```




```
20220 IFPEEK(251)=152THENR=-22:GOTO20300
20230 IFPEEK(251)=28THENR=1:GOTO20300
20240 IFPEEK(251)=148THENR=22:GOTO20300
20250 IFPEEK(251)=140THENR=-1:GOTO20300
20260 GOTO20500
20300 IFPEEK(X+R)=32THEN20400
20310 IFPEEK(X+R)=81THENFORI=135TO165:POKE36876,I:NEXT:POKE36876,0:GD=GD+1
20315 IFPEEK(X+R)=81ANDGD=10THENGOTO9
20316 IFPEEK(X+R)=81THEN20400
20320 POKE36878,10:FORI=135TO195:POKE36877,J:NEXT:POKE36877,0
20330 GOTO10000
20400 POKEX,32:X=X+R:POKE36876,135:POKE36876,138:POKE36876,0:POKEX,87
20500 K=SGN(X-Y):IFABS(X-Y)>QQTHENK=K+K
20505 K=K+3
20506 Y1=Y+W(K)
20507 IFPEEK(Y1)=87THEN20600
20510 IFPEEK(Y1)=32THEN20600
20530 QQ=RND(1)*Q+1:K=K+1:IFK>6THENK=1
20540 GOTO20506
20600 POKEY,32:Y=Y1:POKEY,88:IFX<>YTHEN20210
20610 GOTO20320
21000 POKE36876,183:FORI=1TO200:NEXT:POKE36876,201:FORI=1TO200:NEXT:POKE36876,20
7
21010 FORI=1TO400:NEXT:POKE36876,0:RETURN
```

Prost

Sie haben vom Grünflächenamt Ihrer Stadt den Auftrag bekommen, in einem abgelegenen Wald die Überreste einer wilden Party zu beseitigen.

Leider sind Sie selber noch nicht ganz nüchtern und können nicht zwischen den Himmelsrichtungen unterscheiden. Für jeden Schritt, den Sie machen, verlieren Sie Punkte. Durch Einsammeln der liegengebliebenen Flaschen können Sie Ihr Punktekonto erhöhen. Abzüge gibt es, wenn Sie einen Baum umlaufen.

Das Spiel ist in 3 Runden unterteilt. Immer wenn Sie auf die Fahne in der rechten unteren Ecke treffen, beginnt eine neue Runde.

Bonus gibt es bei Erreichen der nächsten Runde und wenn alle Flaschen aufgesammelt wurden. Das Spiel ist deswegen interessant, weil es viele Spielmöglichkeiten zuläßt und weil man nach der Jagd auf den Highscore auch ein wenig nachdenken muß.

Zum Schluß wird man mit einer Siegermelodie verwöhnt und kann seinen Namen eintragen.

Eine Spielversion für den VC-20 ohne Speichererweiterung.

```
5 DIMME(19),FA(19)
10 POKE36879,24
12 BZ=7702:BF=38422:HI=0:LA=36878:S1=36874
14 PRINT"▒"
15 PRINT"▒▒▒██▒  █ ▒  █ ▒  █ ▒  █ ▒  █"
16 PRINT"██▒ █ ▒ █ ▒ █ ▒ █ ▒ █ ▒ █ ▒ █    ▒ █"
17 PRINT"██▒   █ ▒   █ ▒ █ ▒ █ ▒   █   ▒ █"
18 PRINT"██▒ █    ▒  █  ▒ █ ▒ █    ▒ █  ▒ █"
19 PRINT"██▒ █    ▒ █ ▒ █ ▒   █ ▒   █   ▒ █"
20 PRINTTAB(19)"▒ █"
21 PRINT"██▒(C) BY A.BACHLER█ ▒ █"
22 PRINT"█▒                     █"
23 PRINT"▒▒██*******************"
24 PRINT"██*                *"
25 PRINT"██* SPIELBEGINN = *"
26 PRINT"██*                *"
27 PRINT"██* ▒SPACE-TASTE█   *"
28 PRINT"██*                *"
29 PRINT"██******************"
30 GOSUB110
31 GETA$:IFA$=" "THEN33
32 GOTO31
33 ZE=0:PU=15:RU=0:POKE36869,255
35 RU=RU+1:PU=PU+5:DF=0:AB=0
36 IFO=2+ZETHENPU=PU+10
37 O=0:ZE=ZE+5:IFRU>3THEN90
```

```
38 PRINT"RUNDE";RU
40 FORI=1TO2000:NEXT:PRINT"":GOSUB76
42 PRINT"";PU::IFPU<10THENPRINT"";TAB(14);" "
44 J9=PEEK(37137):POKE37154,127:J8=PEEK(37152):POKE37154,255
46 IF(J9AND8)=0THENGOSUB64:DF=DF-22:IFDF<0THENDF=0
48 IF(J9AND4)=0THENGOSUB64:DF=DF+22:IFDF>462THENDF=462
50 IF(J9AND16)=0THENGOSUB64:AB=AB+1:IFAB>21THENAB=21
52 IF(J8AND128)=0THENGOSUB64:AB=AB-1:IFAB<0THENAB=0
54 IFBZ+DF+AB=8185THEN35
56 IFPEEK(BZ+DF+AB)=2THENPU=PU-5:GOSUB68:IFPU<0THENGOTO90
58 IFPEEK(BZ+DF+AB)=1THENPU=PU+8:O=O+1:GOSUB72
59 IFO=4+ZETHENPU=PU+10
60 POKEBZ+DF+AB,0:POKEBF+DF+AB,6
62 GOTO42
64 PU=PU-1:POKELA,15:POKES1,201:FORI=1TO50:NEXT:POKELA,0:POKES1,0:IFPU<0THENGOTO
90
65 POKEBZ+DF+AB,32:RETURN
68 POKELA,15:FORL=148TO200STEP.7:POKES1+2,L:NEXT
69 FORL=200TO128STEP-1:POKES1+2,L:NEXT
70 POKELA,0:POKES1+2,0:RETURN
72 POKELA,15:FORL=1TO30:FORM=250TO240STEP-1:POKES1+2,M:NEXTM
74 POKES1+2,0:NEXTL:POKELA,0:RETURN
76 PRINT"PUNKTE:"
77 FORH=1TO60+ZE:BA=INT(RND(1)*482)+1:POKEBZ+BA,2:POKEBF+BA,5:NEXT:FORH=1TO2+ZE
78 FL=INT(RND(1)*462)+1:POKEBZ+FL,1:POKEBF+FL,6:NEXT:POKEBZ+483,3:POKEBF+483,2
79 RETURN
90 POKE36869,240:PRINT"SPIELENDE!"
92 PRINT"IHRE PUNKTZAHL:"
93 PRINT"";PU
94 IFPU>HITHENHI=PU:GOSUB103:INPUT"NAME";N$
96 PRINT"BESTE PUNKTZAHL:":PRINT"";N$,HI
98 PRINT"NOCHEINMAL? (J/N)"
99 GETA$:IFA$=""THEN99
100 IFA$="J"THEN33
101 IFA$="N"THENSYS64802
102 GOTO99
103 POKELA,15:FORI=1TO19:POKES1,ME(I):FORH=1TOFA(I):NEXTH,I:POKELA,0:POKES1,0:RE
TURN
110 POKE52,28:POKE56,28
111 AH=7168:FORB=0TO511:POKEAH+B,PEEK(32768+B):NEXT
112 FORB=0TO31:READC:POKEAH+B,C:NEXT
113 FORI=1TO19:README(I),FA(I):NEXT
120 DATA60,90,219,255,219,195,126,60
125 DATA24,24,24,60,60,60,60,60
130 DATA24,60,126,24,60,126,24,24
135 DATA8,24,56,120,248,8,8,28
140 DATA215,50,215,150,215,100,225,400,225,400,228,400
145 DATA228,400,235,600,231,200,225,200,,100,235,100
150 DATA231,300,225,100,219,400,232,800,228,300,223,100,225,800
155 RETURN
```

Fortsetzung von Seite 4
vom Stack und gibt sie auf dem Schirm aus. Ein nochmaliges Aufrufen des Befehles >"."< bewirkt eine Fehlermeldung, da der Stack keine Zahl mehr beinhaltet. Wichtig ist, daß die einzelnen Worte und Zahlen immer durch mindestens ein Leerzeichen getrennt werden, da der Computer das Wort sonst nicht erkennen kann. Nach erfolgreicher Ausführung eines Wortes meldet sich der Computer mit „OK", dies entspricht in BASIC der „READY"-Meldung.
FORTH ist eine sehr nachsichtige Sprache. Alles was eingegeben und nicht direkt als falsch erkannt wird, wird angenommen und eventuell sogar ins Wörterbuch eingetragen. Man darf also nicht blindlings drauflosprogrammieren, sondern man sollte sich überlegen, was und wie man es programmiert.
Bei der Beschreibung des Wörterbuches und der Operationen mit dem Stapelspeicher (Stack) werden Bezeichnungen verwendet, wie sie im ⟨Installation Manual⟩ für *FORTH* festgelegt sind. Diese Bezeichnungen sollten Sie sich einprägen, da sie immer wieder verwendet werden.
Diese Bezeichnungen sind:

adr	Speicheradresse
b	8 bit Byte (die oberen 8 Bit sind null)
c	7 bit ASCII-Zeichen
d	32 bit doppelt lange Festkommazahl mit Vorzeichen
f	Boolsche Variable Ø = „false" (falsch)
n	16 bit-Zahl mit Vorzeichen
u	16 bit-Zahl ohne Vorzeichen
⟨RETURN⟩	RETURN-Taste

Bei der Beschreibung eines Wortes wird künftig folgendes Schema angewendet:
WORT ⟨RETURN⟩ Anzeige des Computers (Stapel vorher --> Stapel nachher)
KOMMENTAR:
Beispiel:
= ⟨RETURN⟩ OK (n1 n2 n3 --⟩ n1 n2)
>n3< wird auf dem Bildschirm ausgegeben
Wie Sie schon wissen, ist >"."< eine *Forth-Anweisung,* die die oberste Zahl des Stapels auf dem Schirm ausgibt. Dies ist also die Print-Anweisung. Vor der Ausführung des Befehles befinden sich 3 Zahlen, nämlich n1, n2, n3 im Stapelspeicher. Davon ist die am weitesten rechts stehende Zahl im TOS (Top of Stack). In diesem Fall also ›n3‹. Nach der Ausführung des Befehles sind nur noch zwei Zahlen (n1, n2) im Stapel, wobei nun ›n2‹ an oberster Stelle steht.
Um noch einmal die Funktion des Stapels zu verdeutlichen, geben Sie bitte folgendes Beispiel ein:
1 2 3 4 5 6 ⟨RETURN⟩ OK (empty --⟩ n1 n2 n3 n4 n5 n6)
Die Zahlen 1 bis 6 wurden nacheinander auf den Stapel gelegt. Um sie wieder abzurufen, geben Sie folgendes ein:
= ⟨RETURN⟩ 6 OK (n1 n2 n3 n4 n5 n6 --⟩ n1 n2 n3 n4 n5)
= ⟨RETURN⟩ 5 OK (n1 n2 n3 n4 n5 --⟩ n1 n2 n3 n4)
= ⟨RETURN⟩ 4 OK (n1 n2 n3 n4 --⟩ n1 n2 n3)
= ⟨RETURN⟩ 3 OK (n1 n2 n3 --⟩ n1 n2)
= ⟨RETURN⟩ 2 OK (n1 n2 --⟩ n1)
= ⟨RETURN⟩ 1 OK (n1 --⟩ empty)
Wie aus dem Beispiel zu erkennen ist, werden die Zahlen in der umgekehrten Reihenfolge ausgegeben. Natürlich ist es gleich, ob man den >"."< Befehl einzeln mit ⟨RETURN⟩ quittiert, oder nur durch ein Leerzeichen trennt.

Arithmetik-Operationen

Die 4 Grundrechenarten sind in *FORTH* und BASIC durch die gleichen Bezeichnungen gekennzeichnet.
+ [n^1 n^2 - -> Summe] Addition
– [n^1 n^2 - -> Differenz] Substraktion
* [n^1 n^2 - -> Produkt] Multiplikation
/ [n^1 n^2 - -> Quotient]Division
Wie bei allen *FORTH-Wörtern,* werden auch hier die Operanten auf dem Stack erwartet, und nach der Ausführung durch das Ergebnis überschrieben.
Einige Beispiele verdeutlichen die Arbeitsweise der Befehlswörter:
123⟨RETURN⟩ OK
[empty - -> n^1 n^2]
+ ⟨RETURN⟩ OK
[n^1 n^2 - -> Summe]
. ⟨RETURN⟩ 15 OK
[Summe - -> empty]
Eine andere Schreibweise:
123 + . ⟨RETURN⟩ 15 OK
43213 + . ⟨RETURN⟩ 256 OK
1000344 + . ⟨RETURN⟩ 1344 OK
500100 50 + + . ⟨RETURN⟩ 650 OK
Die Aufgaben 14–4, 12 * 3 und 2000/50 werden in *FORTH* wie folgt gelöst:
144 – . ⟨RETURN⟩ 10 OK
123 * . ⟨RETURN⟩ 36 OK
2000 50 / . ⟨RETURN⟩ 40 OK
Alle oben aufgeführten Wörter, sind INTEGER-Operationen. Dies bedeutet, daß die Operanten sowie das Ergebnis intern, 16 Bit belegen. Für die Anwendung ergibt sich somit folgender Zahlenbereich:
–32768 <- - - - - - - - - - - 0 - - - - - - - - - - -> 32767
Dieser Zahlenbereich darf nicht überschritten werden, da sich sonst das Ergebnis entsprechend verfälscht. Gleitkommazahlen sind in *FORTH* unzulässig.
Folgende Beispiele zeigen die Grenzen der Integer-Arithmatik:
10000 4 * . ⟨RETURN⟩ –25536 OK
Richtig wäre 10000
32767 1 + . ⟨RETURN⟩ –32768 OK
Richtig wäre 32768
5 2 / . ⟨RETURN⟩ 2 OK
Richtig wäre 2.5
11 2 / . ⟨RETURN⟩ 5 OK
Richtig wäre 5.5
Einige Basic-Freunde werden sich jetzt fragen: »Was soll ich mit einer Programmiersprache die nicht einmal richtig 5 durch 2 teilen kann?« Diese Frage ist sicher nicht ganz unberechtigt, jedoch sollte man bedenken, daß in sehr vielen Spiel-, Grafik-, Steuerungs- und Rechenprogrammen eine Integer-Arithmetik völlig ausreicht. Eine Gleitkommazahl können wir durch Wahl einer anderen Einheit in eine Integer-Zahl umwandeln. Beispielsweise können wir den Betrag 1,25 DM in 125 Pf umwandeln.
In einer der nächsten Folgen werden noch einige 32 Bit Arithmetik-Befehle vorgestellt. Diese erlauben durch geschickte Programmierung eine höhere Genauigkeit, als sie in Basic vorhanden ist.
Die *FORTH-Schreibweise* verschiedener Rechenaufgaben:
Aufgabe: 4 + ⟨17 * 12⟩
17 12 * 4 + . ⟨RETURN⟩ 208 OK
Durch das Stack-Konzept werden Klammern überflüssig.
Aufgabe: ⟨3 + 9⟩ * ⟨4 + 6⟩
3 9 + 4 6 + * . ⟨RETURN⟩ 120 OK
Aufgabe: 45 / 5 * 100
45 5 / 100 * . ⟨RETURN⟩ 900 OK
Weitere Arithmetik Befehle sind:
MOD [u^1 u^2 - -> Rest]
Dividiere u^1 durch u^2 und lege Rest auf Stack
11 3 MOD .⟨RETURN⟩ 2 OK
8 10 MOD .⟨RETURN⟩ 8 OK
/ MOD [u^1 u^2 - -> Rest Quotient]
Dividiere u^1 durch u^2 und lege Rest sowie Quotient auf Stack
11 3 / MOD .⟨RETURN⟩ 3 2 OK
8 10 / MOD .⟨RETURN⟩ 0 8 OK
* / [n^1 n^2 n^3 - -> n^4]
Errechne den Verhältnis-Faktor $n^4 = n^1 * n^2 / n^3$. Das Produkt $n^1 * n^2$, ist als Zwischenergebnis 32 Bit lang. Hierdurch ist eine höhere Rechengenauigkeit gewährleistet, als durch die Se-

quenz n^1 n^2 * n^3 / erreichbar wäre. Diese Anweisung wird häufig zum Rechnen mit genauen Konstanten verwendet. Folgende Konstanten können durch das Verhältnis n^2 / n^3 errechnet werden:

Konstante	Richtiger Wert	Verhältnis n^2 / n^3
PI	3.1415926	355 / 113
Wurzel aus 2	1.4142135	19601 / 13860
Wurzel aus 3	1.7320508	18817 / 10864

Einige Anwendungsbeispiele:

10 355 113 * / . ⟨RETURN⟩ 31 OK
Entspricht 10 * PI
100 355 113 * / . ⟨RETURN⟩ 314 OK
Entspricht 100 * PI

Wie aus dem Beispiel zu erkennen ist, wird mit diesem Befehl die größte Genauigkeit, die Integer-Zahlen bieten, erzielt.

* / MOD [n^1 n^2 n^3 – –> Rest n^4]
Dieser Befehl entspricht dem Vorangehenden, mit der Ausnahme, daß auch der Rest der Division auf den Stack gelegt wird.

10 355 113 * / MOD . .
⟨RETURN⟩ 31 47 OK
100 355 113 * / MOD . .
⟨RETURN⟩ 314 18 OK

ABS [n^1 – –> u^1]
Dieser Befehl wandelt n^1 mit Vorzeichen in die vorzeichenlose Zahl u^1 um. In Basic gibt es den gleichen Befehl.

– 123 ABS .⟨RETURN⟩ 123 OK
– 1 ABS . ⟨RETURN⟩ 1 OK
5 ABS . ⟨RETURN⟩ 5 OK

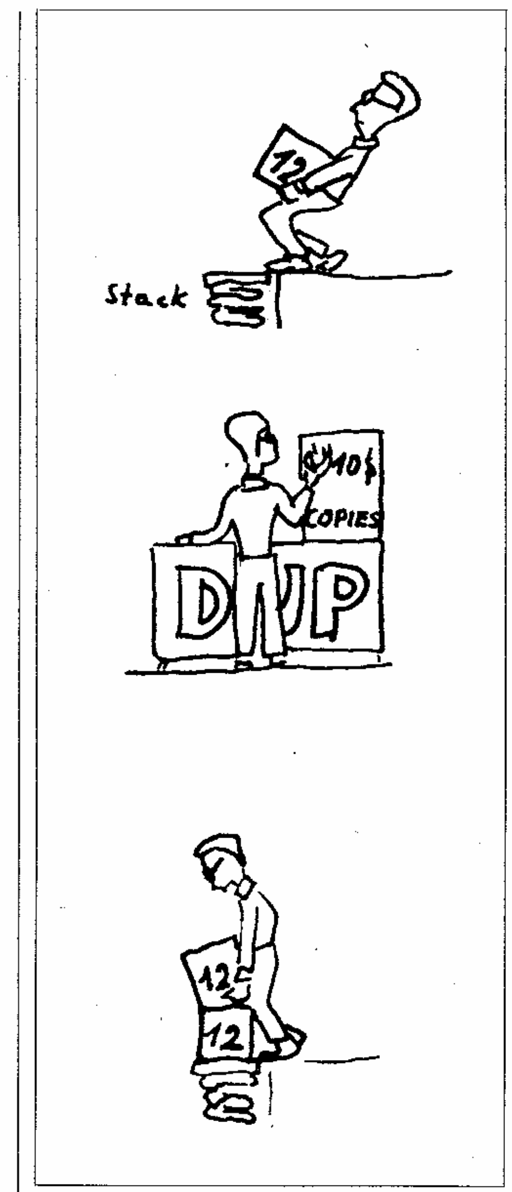
Häufig benutzte Befehls-Folgen, wurden in *FORTH* zu einem Befehl zusammengefügt:

1 +	[n^1 – –> n^2]	entspricht der Befehlsfolge 1 +
1 –	[n^1 – –> n^2]	entspricht der Befehlsfolge 1 –
2 +	[n^1 – –> n^2]	entspricht der Befehlsfolge 2 +
2 –	[n^1 – –> n^2]	entspricht der Befehlsfolge 2 –
2 *	[n^1 – –> n^2]	entspricht der Befehlsfolge 2 *
2 /	[n^1 – –> n^2]	entspricht der Befehlsfolge 2 /

Diese Befehls-Kombinationen sind nicht in allen *FORTH-Versionen* enthalten, können jedoch sehr leicht definiert werden. Das Definieren von Befehlen wird Thema des nächsten Teiles werden.

Stack Manipulation

Wir wir schon bei einigen Befehlen bemerkt haben, ist es wichtig, daß die Parameter eines Befehls immer in der richtigen Reihenfolge auf dem Stack liegen. Bei umfangreichen Befehlsfolgen muß deshalb oft die Reihenfolge



der Operanten geändert werden, um eine anschließende Operation zu ermöglichen. Um dies zu erreichen, werden folgende Befehle verwendet:

[n^1 – –> n^1 n^1]
DUP ist der am häufigsten verwendete Befehl. Man kann ihn sich sehr leicht merken, da DUP für Duplizierung steht. Er hat eine einfache, jedoch wichtige Aufgabe. Dieser Befehl dupliziert die oberste Zahl des Stacks.

100 DUP . . . ⟨RETURN⟩ 0 0 10 OK
12 DUP . . ⟨RETURN⟩ 12 12 OK

– DUP [n^1 – –> n^1 n^1]
Dieser Befehl entspricht dem DUP-Befehl, mit der Ausnahme, daß zuvor getestet wird, ob die zu duplizierende Zahl von Null verschieden ist. Nur wenn dies der Fall ist, erfolgt ihre Duplizierung.

100 – DUP . . ⟨RETURN⟩ 0 10 OK
12 – DUP . . ⟨RETURN⟩ 12 12 OK

DROP [n^1 – –> empty]
ist das einfachste Befehlswort, welches lediglich die oberste Zahl des Stacks löscht.

100DROP . ⟨RETURN⟩ 10 OK
12 DROP . ⟨RETURN⟩ Fehlermeldung: Stack leer

SWAP [n^1 n^2 – –> n^2 n^1]
Dieser Befehl verändert ganz einfach die Anordnung der beiden obersten Zahlen. Dies ist oft vor Arithmetik-Operationen erforderlich.

100 SWAP . .⟨RETURN⟩ 100 OK
334 SWAP . .⟨RETURN⟩ 334 OK

OVER [n^1 n^2 – –> n^1 n^2 n^1]
Dieser Befehl arbeitet ähnlich wie der DUP-Befehl, jedoch wird hier der »Second« [d. h. die vorletzte Eintragung in den Stack] kopiert.

100 OVER . . ⟨RETURN⟩ 10 0 10 OK
334 OVER . . ⟨RETURN⟩ 33 4 33 OK

ROT [n^1 n^2 n^3 – –> n^2 n^3 n^1]
Hiermit wird die Anordnung der höchsten drei Stackeintragungen in zyklischer Rotation verändert. Dabei landet der älteste Eintrag an oberster Stelle.

10 20 30 ROT . . .
⟨RETURN⟩ 10 30 20 OK
10 20 30 ROT ROT . . .
⟨RETURN⟩ 20 10 30 OK
10 20 30 ROT ROT ROT ROT . .
⟨RETURN⟩ 30 20 10 OK

Einige erweiterte *FORTH-Versionen*, wie *64 FORTH* oder *FORTH 79*, besitzen noch einige, zusätzliche Stack-Manipulations-Befehle. Einen gewissen Standard besitzen folgende Befehle:

DEPTH [empty – –> n^1]
Es wird die Anzahl der auf dem Stack liegenden Parameter übergeben.

10 20 DEPTH . . .
⟨RETURN⟩ 2 20 10 OK
DEPTH .⟨RETURN⟩ 0 OK

PICK [n^1 – –> n^2]
Die »n-te« Zahl wird auf den Stack gelegt. 1 PICK entspricht DUP; 2 PICK entspricht OVER

12 43 4 33 4 PICK . . . .
⟨RETURN⟩ 12 33 34 4 12 OK
12 3 2 PICK . . .
⟨RETURN⟩ 12 3 12 OK

ROLL [n – –>]
Dieser Befehl rotiert die »n-te« Zahl des Stacks an die oberste Stelle. 2 ROLL entspricht SWAP; 3 ROLL entspricht ROT.

12 13 14 15 4 ROLL . . . .
⟨RETURN⟩ 12 15 14 13
12 13 14 15 3 ROLL . . . .
⟨RETURN⟩ 13 15 14 12



Definition neuer Befehls-Worte
Nachdem wir uns mit den Arithmetik-Operationen beschäftigt haben, wollen wir uns nun mit der Definition neuer Befehle befassen. Die bislang vorgestellten *FORTH-Befehle* sind bis jetzt ausschließlich interaktiv und interpretativ aufgerufen worden. Das bedeutet, daß ein Wort (z. B. DUP oder +) mit der Eingabe seines Namens in die Tastatur aufgerufen wird, indem der äußere Interpreter dann das Wörterbuch (Dictionary) nach diesem Wort absucht, es lokalisiert und ausführt. Diese Art der Befehls-Abarbeitung wird im Gegensatz zu Basic nur bei der direkten Eingabe von Befehlen verwendet. Wir haben nun bereits so viele Befehle kennengelernt, daß wir daran gehen können, kleinere Programme damit zu erstellen. Wie schon im ersten Teil erwähnt, ist das gleichbedeutend damit, die zu anfangs erwähnten Befehle zu neuen Befehlen zusammenzusetzen.

Wollen wir einen Befehl definieren, so müssen wir als erstes mit dem Befehl »:«, den *FORTH-COMPILER* einschalten. Als nächstes wird die Bezeichnung und danach die eigentliche Definition des neuen Befehls eingegeben. Mit dem Befehl »:« wird der *FORTH-COMPILER* wieder ausgeschaltet und damit die Definition abgeschlossen.
Wir wollen nun einen Befehl definieren, welcher eine Zahl mit sich selbst multipliziert und das Ergebnis auf den Stack legt. Als Bezeichnung wählen wir »QUADRO«.

: QUATRO DUP ☆ ; ⟨RETURN⟩ OK

Wie zu erkennen ist, besteht die eigentliche Definition aus einer unbegrenzten Zahl von Befehlen. Der Befehl DUP dupliziert die oberste Zahl des Stack und der nachfolgende »☆« Befehl bildet, durch die Multiplikation, die Quadratzahl.

In den nachfolgenden Beispielen können wir den neuen Befehl testen:

2 QUATRO · ⟨RETURN⟩ 4 OK
5 QUATRO · ⟨RETURN⟩ 25 OK
10 QUATRO · ⟨RETURN⟩ 100 OK

Durch den Befehl »VLIST« können wir das gesamte Wörterbuch (Befehl's-Liste) auf Drucker oder Monitor ausgeben lassen. Der neue Befehl »QUATRO« steht nun am Anfang dieser Liste.

Bei der Definition von neuen Befehlen muß darauf geachtet werden, daß die Befehlsbezeichnung keine 31 Zeichen übersteigt. Es ist nicht unbedingt notwendig und auch nicht immer möglich, daß die Definition in eine einzige Zeile geschrieben wird. Als Beispiel wollen wir einen Befehl definieren, welcher die Kubikzahl eines Operanten errechnet:

: KUBIK DUP DUP ☆ ☆ ; ⟨RETURN⟩ OK

Wir können das gleiche auch wie folgt eingeben:

```
: KUBIK        ⟨RETURN⟩
   DUP DUP     ⟨RETURN⟩
   ☆ ☆         ⟨RETURN⟩
   ;           ⟨RETURN⟩ OK
```

Bei der direkten Eingabe von Befehlen ist die Schreibweise kaum von Bedeutung, da die Struktur der Eingabe nicht gespeichert wird. Ganz anders ist dies bei der Programmierung mit dem Texteditor. Hier sollte man unbedingt auf Strukturierung und gute Dokumentation achten, um auch später noch Erweiterungen oder Änderungen vornehmen zu können. Wir werden uns später noch genauer mit diesem Thema auseinandersetzen.
Nun wollen wir auch unseren KUBIK-Befehl testen:

2 KUBIK · ⟨RETURN⟩ 8 OK
5 KUBIK · ⟨RETURN⟩ 125 OK
10 KUBIK · ⟨RETURN⟩ 1000 OK

Werden in einem Programm oft die gleichen Befehls-Folgen verwendet, so ist es sinnvoll, diese zu einem Befehl zusammenzufassen:

: 1+ 1+ ; ⟨RETURN⟩ OK
: 1– 1– ; ⟨RETURN⟩
: 2+ 2+ ; ⟨RETURN⟩

Einige dieser einfachen Befehls-Folgen sind im *FORTH-Standard* schon enthalten und wurden im letzten Teil schon vorgestellt. Es ist nun sehr leicht, die fehlenden Kombinationen zu definieren. Wird in *FORTH* ein Befehl zweimal definiert, so gilt immer die letzte Defination.
FORTH-Befehle können auch zur Umsetzung dienen. Bei den meisten farbigen Computern dient ein Zahlencode zur Bestimmung einer Farbe.
Farbcode des Commodore 64:
0 = Schwarz
1 = Weiß
2 = Rot
3 = Türkis
usw.
Wollen wir statt des umständlichen Codes einfach die Farbe eingeben, können wir dies durch die folgenden Definitionen erreichen:

```
: SCHWARZ 0 ;  ⟨RETURN⟩
: WEISS   1 ;  ⟨RETURN⟩
: ROT     2 ;  ⟨RETURN⟩
: TÜRKIS  3    ⟨RETURN⟩
```
usw.

Die Funktion ist sehr einfach: Nach Eingabe der Farbbezeichnung legt der entsprechende Befehl den dazugehörigen Wert auf den Stack.

Es ist auch möglich, mit den Farben zu rechnen:

WEISS ROT + . ⟨RETURN⟩ 3 OK

Mit Hilfe eines PRINT-Befehls können wir auch leicht eine Vokabel-Übersetzung realisieren. Der PRINT-Befehl besitzt in *FORTH* folgende Syntax: ."text".

Elementare String-Befehle
Leider existiert in *FORTH* nur ein kleiner Befehlssatz zur Stringverarbeitung. Der *FORTH-Kern* enthält jedoch einige Worte für einfache Ein- und Ausgaben von Strings (Texten), die für viele Applikationen völlig ausreichen dürften.
»CR«
Dieser Befehl bewirkt auf dem Ausgabegerät (normal, Monitor) Zeilenvorschub und Wagenrücklauf.
12 13 14 . . CR . (return) 14 13
12 OK
. "TRONIC· CR . „VERLAG" (RETURN) TRONIC
Verlag OK
SPACE
Ein SPACE-Zeichen wird ausgegeben
. „APRIL" SPACE . „MAI" (RETURN)
APRIL MAI OK
Spaces (n1 –» empty)
Eine bestimmte Anzahl von SPACE-Zeichen wird ausgegeben.
. „APRIL" 4 SPACE . „MAI" (RETURN) APRIL MAI OK
EMIT (c –» empty)
Der ASCII-Charakter c wird ausgegeben. In Basic lautet dieser Befehl CHR$ (x).
65 EMIT (RETURN) A OK
66 EMIT (RETURN) B OK
EXPECT (adr n –» empty)
Dieser Befehl entspricht in etwa dem INPUT des Basic-Interpreters. Ein String mit ,n'-Zeichen wird von der Tastatur erwartet und ab Adresse ,adr' abgespeichert. Ein RETURN kann diese Prozedur beenden, auch wenn die Anzahl ,n' noch nicht erreicht sein sollte.
10000 11 EXPECT (RETURN)
COMPUTRONIC (RETURN) OK
TYPE (adr n –» empty)
Dieser Befehl gibt eine Zeichenkette, die als Adresse ,adr' gespeichert und ,n'-Zeichen lang ist, auf dem Ausgabegerät aus.
10000 11 TYPE (RETURN) COMPUTRONIC OK
COUNT (adr1 –» adr2 n)
Dieser Befehl erwartet eine Adresse ,adr 1' auf dem Stack und ermittelt hieraus die Startadresse und die Zeichenzahl des Strings. Dies sind die Angaben, die von Type erwartet werden.

Die Anwendung dieses Wortes erfolgt also üblicherweise in der Form ‚adr1 COUNT TYPE',
wobei als ‚adr1' die Adresse des COUNT-Bytes (Zeichenanzahl) übergeben wird. Der eigentliche String muß dem COUNT-Byte folgen. KEY (-» c)
Dieser Befehl wartet auf eine beliebige Tastatureingabe und übergibt den entsprechenden ASCII-Code auf den Stack. Bei einigen Forth-Versionen ist die RETURN-Taste zur Quittierung erforderlich.

KEY . (RETURN)
A
65 OK
KEY KEY . . (RETURN)
A B
66 65 OK
KEY EMIT (RETURN)
D
D OK

Forth-Editor

Wir haben das Definieren von neuen Befehlswörtern kennengelernt. Wir haben auch gelernt, daß in *FORTH* definieren gleichbedeutend mit Programmieren ist. Um jedoch größere Programme entwickeln zu können, müssen wir diese abspeichern, laden und editieren können. Dies geschieht in *FORTH* – im Gegensatz zu Basic – mittels Texteditor. Der Texteditor ist leider in jeder *FORTH*-Version unterschiedlich zu handhaben. Wir wollen uns deshalb mit unserer Beschreibung auf den weitverbreiteten *„FIG-STANDARD"* beziehen.
Im Gegensatz zu vielen anderen Texteditoren wird in *FORTH* immer nur eine Bildschirmseite editiert. Man spricht dabei von einem „SCREEN". Diese SCREENs werden mit einer Nummer versehen und in der Regel auf Diskette gespeichert. Je nach *FORTH-Version* können 20 bis über 100 solcher Screens abgespeichert werden. Ein solcher Screen ist in 16 Zeilen zu je 64 Zeichen aufgeteilt. Da bei einigen Computern nur 40 Zeichen darstellbar sind, wird durch ein Links/rechts-Scrollen ein größerer Bildschirm simuliert. Ein Beispiel dafür ist *„64 FORTH"* von HES. Die meisten *FORTH-Versionen* besitzen mehrere TEXTPUFFER. Diese erlauben das gleichzeitige Einladen mehrerer „SCREENs".
Wollen wir einen Screen auf dem Bildschirm ausgeben lassen, so können wir dies mit dem Befehl: LIST.
Die Zeilennummern dienen nur zur Orientierung und werden deshalb nicht von allen *FORTH-Versionen* ausgegeben. Die erste Zeile sollte immer einen Kommentar enthalten, um nicht den Überblick zu verlieren. Der Kommentar beginnt mit einem (Zeichen, gefolgt von einem Leerzeichen und geht bis zum Zeichen). In diesem Kommentar sollte das Datum und die Aufgabe des SCREENs beschrieben werden. Selbstverständlich können auch in anderen Zeilen noch Kommentare notiert werden.
Wir haben nun gesehen, wie ein beschriebener SCREEN aussehen kann und wollen nun daran gehen, einen SCREEN zu beschreiben.
Zuerst suchen wir uns mit Hilfe eines Handbuches einen freien SCREEN aus. Mit . . . LIST können wir uns überzeugen, daß der SCREEN leer ist. Ist beispielsweise der SCREEN Nummer 10 unbeschrieben, so geben wir folgenden Befehl ein: 10 EDIT (RETURN).

Nun befinden wir uns im eigentlichen Editor. Dieser ist im FIG-STANDARD zeilenorientiert. Das bedeutet, daß der Text mit Hilfe mehrerer Kurzbefehle editiert bzw. erstellt wird.

Die folgende Aufstellung zeigt die Funktion der Standard-Befehle:
LINE n --» adr

Dieser Befehl übergibt dem Stack die Adresse der Zeilennummern. Ist der



entsprechende Screen nicht im Puffer, so wird dieser von der DISK eingelesen.
– MOVE adr – – » n
Die Textzeile ab Adresse adr wird in Zeile n kopiert.
H n – – »
HOLD übernimmt die Textzeile n nach PAD und hält sie dort zur Ausgabe bereit.
E n – – »
ERASE löscht Zeile Nummer n und füllt diese mit Leerzeichen (Blanks) aus.
S n – – »
Dieser Befehl fügt vor Zeile n eine Leerzeile ein. Nachfolgende Zeilen werden um eine Zeile nach unten verschoben. Die unterste Zeile geht verloren.
D n – »
Die Zeile n wird gelöscht. Dadurch wird der nachfolgende Text um eine Zeile angehoben.
T n – – »
Zeile Nummer n wird auf dem Schirm ausgegeben.
L n – – »
Der zur Zeit bearbeitete Screen wird ausgelistet.
R n – – »
Dieser Befehl ist das Gegenstück zum H-Befehl. Die im PAD abgelegte Zeile wird in Zeile n ausgegeben.
P n – – »
Der folgende Text wird in Zeile n übertragen. Der alte Zeileninhalt wird überschrieben.
I n – – »
Der PAD wird ab Zeile n ausgegeben. Die nachfolgenden Zeilen werden um eine Zeile nach unten verschoben.
CLEAR n – – »
Der Screen n wird mit Leerzeichen (Blanks) vollgeschrieben und somit gelöscht.
COPY n1 n2 – – »
Dieser Befehl kopiert den gesamten Screen n1 nach Screen n2.
Die nachfolgend aufgestellten Befehle gehören zwar nicht zum Editor, sind jedoch zum Verwalten der Screens erforderlich.
LIST n1 – – »
Screen n1 wird auf dem Schirm ausgelistet.
FLUSH
Alle geänderten Screens werden auf Diskette abgespeichert.
EDIT n1 – – »
Screen n1 wird eingeladen und kann editiert werden.
LOAD n1 – – »
Dies ist der wichtigste Befehl. Screen n1 wird in den Puffer eingeladen und dann Zeile für Zeile kompiliert. Wird ein Wort mehrmals definiert, so wird dies mit einer Meldung angezeigt. Der Kompilier-Vorgang wird in diesem Falle jedoch nicht unterbrochen. Um jedoch unnötigen Speicherverlust zu vermeiden, sollte man das Wörterbuch vor jedem neuen Kompilier-Vorgang zurücksetzen. Wie wir wissen, wird dies mit dem Befehl FORGET . . . gemacht.

Sicher wird sich so mancher schon gefragt haben, wie er längere Programme schreiben soll, wenn doch immer nur ein Screen kompiliert wird. Des Rätsels Lösung ist ganz einfach:
Mit Hilfe des – – »-Befehls kann man beliebig viele Screens miteinander verbinden und somit sehr lange Programme schreiben.

Der – – »-Befehl wird in diesem Fall an das Ende des letzten Screen's gesetzt. Aufgerufen wird jeweils nur der erste Screen. Je nach Datenträger und Anzahl der Puffer dauert der Kompiliervorgang bis zu einigen Minuten.

Einige *Forth-Versionen* besitzen auch die Möglichkeit, das Wörterbuch direkt abzuspeichern.

Im *C64 FORTH* von *HES* heißt der Befehl zum Abspeichern des Wörterbuches: *DSAVE*.

Das Einladen geschieht in diesem Fall mit DLOAD. Das D steht in beiden Fällen für Dictionary und ist gleichbedeutend mit Wörterbuch. Bei der Benutzung dieser beiden Befehle sollte man bedenken, daß nur der compilierte Code angespeichert wird und somit keine Änderungen mehr möglich sind. Wir haben nun eine Menge über den *FORTH-Texteditor* erfahren und können somit auch längere Programme schreiben, ohne diese nach dem Ausschalten zu verlieren.

Wie schon erwähnt, ist es von größter Wichtigkeit, seine *FORTH-Programme* gut zu dokumentieren, da *FORTH-Befehle* allein sehr unübersichtlich wirken.

Zur Dokumentation ist übrigens an dieser Stelle zu sagen, daß Kommentare unbestreitbar den Vorzug haben, unverrückbar physikalisch an die kommentierte Definition geknüpft zu sein. Daher wird jede Änderung eines Programms auch mit der Aktualisierung der Kommentierung zwangsläufig verbunden sein. Dennoch sollte man die Kommentare auf der Diskette knapp halten. Genauer:
Die mit dem Code verknüpften Kommentare sollten knapp sein, da sonst die Kompilierungszeit unnötig erhöht wird.
Daneben ist die Lesbarkeit von Texten innerhalb von Codes relativ schlecht und verschlechtert auch die Lesbarkeit des Codes.
Da die Diskettenlaufwerke von Heimcomputern auch keine unbegrenzte Speicherkapazität besitzen, sollte man sich überlegen, ob man die Dokumentation nicht wie bei Assembler-Programmen getrennt vom Programm vornimmt.
Wie wäre es denn z. B. mit Papier und Bleistift?

Bedingte Verzweigungen
Das Forth-Befehlswort „IF" eröffnet alle bedingten Verzweigungen in FORTH-Programmen. Dieses Befehlswort erwartet eine 16-Bit Zahl auf dem Stack und behandelt dieses als Flag. Der „IF"-Befehl arbeitet nur zusammen mit dem Befehlswort „ENDIF". „ELSE" ist ebenfalls möglich, jedoch nicht unbedingt nötig. Folgendes Beispiel zeigt den Zusammenhang der einzelnen Befehle:

```
IF      (IF testet das Flag; falles es
         wahr (= 1) ist ...)

...     (werden die nebenstehenden
         Anweisungen ausgeführt

...     (ansonsten)

ENDIF(fortgefahren)

IF
...     (falls Flag = 1, werden neben-
         stehende Worte)

...     (bis ELSE ausgeführt)

ELSE
...     (andernfalls erfolgt die Ausfüh-
         rung der Worte)
...     (zwischen ELSE und ENDIF)

ENDIF
```

Bei der Verwendung von Verzweigungen sollte man immer bedenken, daß diese nicht interprevativ verwendet werden dürfen, also nur innerhalb einer Definition. Probieren wir doch einmal die folgende, einfache Definition:

```
:DIVISION(Division mit Kommentar,
          falls Rest nicht Null) (Para-
          meter wie bei / – Befehl)

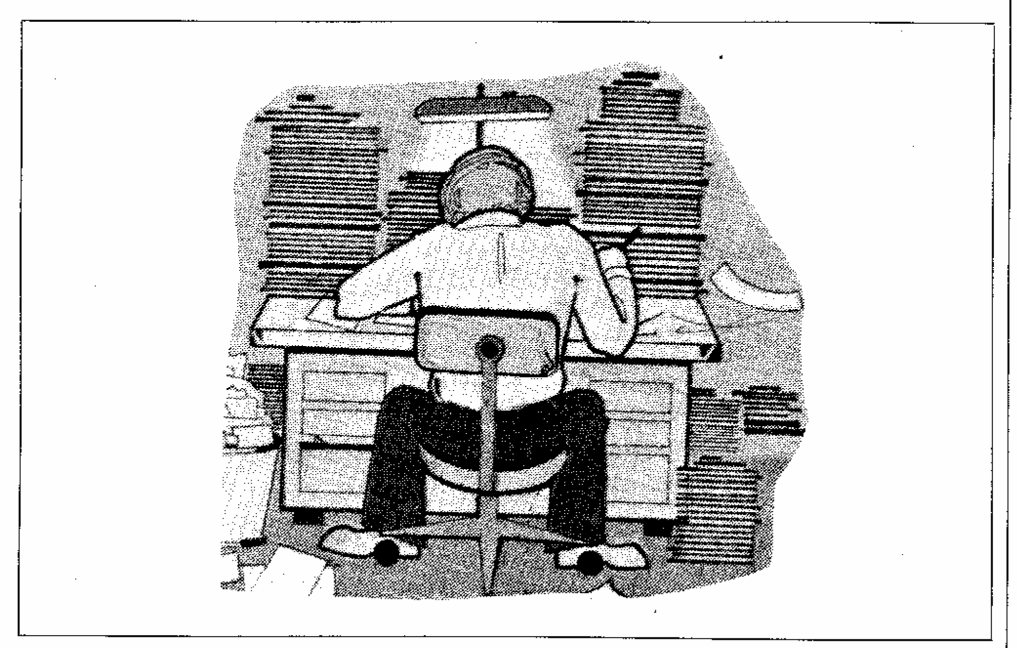
/MOD.     (Division mit Rest)

SWAP      (Bringe Rest zum TOS und
          teste ihn als Flag)

„IFDIVISION geht nicht auf"

ENDIF
;         (falls Rest = „wahr", wird
          der Text gedruckt)
```

Das nun definierte Wort „DIVISION“ erwartet also zwei Zahlen auf dem Stack, dividiert diese und beläßt den Quotienten auf dem TOS. Die Syntax des neuen Befehls entspricht genau dem Original "/" Befehl. Im Gegensatz zu diesem, wird jedoch bei Entstehung eines Restes die Meldung „DIVISION GEHT NICHT AUF“, ausgegeben.
Wir wollen nun ein Beispiel zeigen, welches den „ELSE“-Befehl vorführt. Es handelt sich hier um einen Vergleichs-Befehl, welcher angibt, ob die erste Zahl kleiner oder größer als die zweite Zahl ist.



```
:VERGLEICHE (Vergleich zweier Zahlen)
<          ."(FLAG ermitteln)
IF         ."KLEINER"
ELSE       ."GROESSER"
ENDIF
;
```

Beispiel:
33 100 VERGLEICHE (RETURN) KLEINER OK
100 33 VERGLEICHE (RETURN) GROESSER OK

Zum Vergleich dasselbe noch einmal in der Programmiersprache Basic:

```
10 INPUT „ZAHL1“;Z1

20 INPUT „ZAHL2“;Z2

30 IF Z1<Z2 THEN PRINT „KLEINER“

40 IF Z1>Z2 THEN PRINT „GROESSER“

50 END
```

Da im Standard-Basic kein „ELSE“-Befehl zugelassen ist, muß man sich mit zwei „IF“ Anweisungen behelfen, wenn man nicht zu dem UNSAUBEREN „GOTO“-Befehl greifen will. Dies geht selbstverständlich auf Kosten der Zeit.
Im übrigen sei an dieser Stelle darauf hingewiesen, daß einige *FORTH-Versionen* statt des Befehles „ENDIF“ den Befehl „THEN“ verwenden. Dies muß in dem entsprechenden Manual nachgelesen werden.
Wir haben nun einiges über die „IF“-Anweisung gelernt und wollen uns nun die eigentlichen Vergleichsbefehle näher ansehen:

< (n1 n2 n3 --> n1 f)
Flag wird 1 wenn die Zahl n2 kleiner als n3 ist.

> (n1 n2 n3 --> n1 f)
Flag wird 1 wenn die Zahl n2 größer als n3 ist.

= (n1 n2 n3 --> n1 f)
Flag wird 1 wenn beide Zahlen gleich sind

0< (n1 n2 n3 --> n1 n2 f)
Flag wird 1 wenn n3 eine negative Zahl ist

0= (n1 n2 n3 --> n1 n2 n3)
Flag wird 1 wenn die Zahl n3 gleich null ist.

Als Ergebnis des Vergleiches wird immer eine Boolsche Zahl abgelegt. Diese wird auch als Flag bezeichnet und gibt an, ob die Aussage richtig oder falsch ist. Ist diese richtig, so wird eine 1, ansonsten eine 0 auf den Stack gelegt.

Beispiele:

```
23 30 < . <RETURN> 1 OK
30 23 < . <RETURN> 0 OK
30 44 = . <RETURN> 0 OK
-2  0 < . <RETURN> 1 OK
```

Bei dem Vergleich zweier Zahlen sollte man bedenken, daß diese vom Vergleichsbefehl überschrieben werden. Will man diese jedoch für weitere Berechnungen erhalten, so kann man dies mit dem Befehl OVER erreichen:
23 30 OVER OVER < . . . <RETURN> 1 30 23

Programmschleife DO...LOOP

Ein wichtiger Bestandteil aller Programmiersprachen sind Schleifen. Unter Schleife versteht man einen oder mehrere Befehle, die durch zwei Kennzeichnungen eingeklammert sind. Diese eingeklammerten Befehle werden beliebig oft durchlaufen. Ist eine bestimmte Anzahl durchlaufen, wird das Programm fortgesetzt. In *FORTH* gibt es genau wie in Pascal keinen „GOTO“ Befehl, deshalb ist der Programmierer auf die Schleife angewiesen.

Der Schleifenbefehl hat in *FORTH* folgende Syntax:

DO ... LOOP (n1 n2 n3 --> n1)
Die Schleife wird von n3 bis n2 in Einerschritten durchlaufen. Die Zahl n3 muß in diesem Fall kleiner als n2 sein.

Die oberste Zahl auf dem Stack wird immer als Anfangswert und die darunterliegende Zahl als Endwert verwendet. Während der Schleife wird der Zahlenparameter immer um eins erhöht, deshalb muß der Endwert immer größer als der Anfangswert sein. Ist der Zahlenparameter größer oder gleich dem Endwert, so wird die Schleife verlassen. Da diese Kontrolle jedoch immer erst am Ende einer Schleife durchgeführt wird, ist die Schleife bereits einmal durchlaufen.
Will man während eines Schleifendurchlaufs den Zahlenparameter auslesen, kann man dies mit Hilfe des Wortes „I“. Dieses Wort legt den Zahlenparameter auf den Stack.

Wie der „IF“ Befehl kann auch die DO-LOOP-Schleife nicht interpretativ ausgeführt werden und muß deshalb innerhalb einer Definition stehen.
Das folgende Beispiel gibt die Zahlen 0–12 auf dem Schirm aus:

```
: ZAHLEN 12 0 DO I . LOOP ; <RETURN> OK

ZAHLEN <RETURN> 0 1 2 3 4 5 6 7 8 9 10 11 12 OK
```

Das gleiche in BASIC:
FOR I = 0 TO 12 : PRINT I, :NEXT 1

Ist der Endwert kleiner als der Anfangswert, so wird die Schleife nur einmal durchlaufen:

: ZAHLEN 0 12 DO I . LOOP ; <RETURN> OK
ZAHLEN <RETURN> 12 OK

Das nächste Beispiel zeigt zwei DO-LOOP-Schleifen die ineinander verschachtelt sind. Der Aufruf der neuen Definition gibt die Zahlen 0 bis 9 in einer Dreieckmatrix aus.

: SCHLEIFE1 0 DO I . LOOP ; <RETURN> OK

: SCHLEIFE2 0 DO CR 1 – DUP SCHLEIFE1 LOOP ; <RETURN> OK
11 10 SCHLEIFE2 <RETURN> OK

0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0 OK

In vielen Programmen werden DO-LOOP-Schleifen zur Verzögerung von Programmabschnitten verwendet. Durch den folgenden Vergleich wird ersichtlich, wie schnell die Programmiersprache *FORTH* arbeitet:

Eine Leerschleife in FORTH:
: PAUSE 10000 0 DO LOOP ;

Die Ausführungszeit dieser Schleife beträgt im Durchschnitt 4 Sekunden. Je nach *FORTH-Version* und Microprozessor kann diese Zeit etwas länger oder auch kürzer werden.

Die gleiche Schleife in BASIC:
10 FOR I = 0 TO 10000 : NEXT 1

In Basic ist die durchschnittliche Ausführungszeit etwa 4 bis 5 mal so lang wie in *FORTH*. Wie zu erkennen ist, eignet sich somit *FORTH* auch für Programme, in denen es auf genaue „Pause-Zeiten" ankommt. (Z. B. Eprom-Programmierung oder sonstige Ein- bzw. Ausgaben).

So mancher wird sich inzwischen fragen, ob man das Zahlenparameter nicht auch um 2, oder einer anderen Zahl erhöhen kann. Dies geschieht in Basic durch den Befehl „STEP".
Dies ist auch in *FORTH* möglich, indem man statt des „LOOP"-Befehls ein „+LOOP" einsetzt. In diesem Fall wird die Zahl, welche vor dem „+LOOP" auf dem Stack liegt, als Increment benutzt. Ist diese Zahl im negativen Bereich, so wird die Schleife bei jedem Durchlauf erniedrigt. In diesem Fall muß der Endwert kleiner als der Anfangswert sein, da sonst nur ein Durchlauf erfolgt.

Die Syntax: +LOOP (n1 n2 n3 --> n1 n2)

Einige Beispiele:

: ZAHLEN 10 0 DO I . 2 +LOOP ; <RETURN> OK

ZAHLEN <RETURN> 0 2 4 6 8 OK
In BASIC:

10 FORI = 0 TO 10 STEP 2 : PRINT I, : NEXT I

Umgekehrte Schleife:

: ZAHLEN 0 10 DO I . –2 +LOOP ; <RETURN> OK

ZAHLEN <RETURN> 10 8 6 4 2 OK

In Basic:

10 FORI = 10 TO 2 STEP –2 : PRINT I, : NEXT I

Bei Verwendung des „+LOOP" sollte man daran denken, daß vor dessen Ausführung eine 0 zur Endlosschleife führt.

BEGIN...UNTIL-Schleife

Da in *FORTH* kein Sprungbefehl vorhanden ist, müssen einige spezielle Schleifen diese ersetzen. Eine davon ist die BEGIN-UNTIL-Schleife.
Diese Schleife wird so lange durchlaufen, bis vor dem Befehl „UNTIL" ein „WAHR-Flag" gesetzt wird. Eine 0 führt zu einem neuen Durchlauf. Diese BEGIN-UNTIL-Struktur kann also sehr leicht den „GOTO"-Befehl ersetzen und führt somit zu einem übersichtlicheren Programm.
Die Struktur der Schleife sieht wie folgt aus:

```
BEGIN
 ...          (Befehle innerhalb der
              Schleife)
 ...

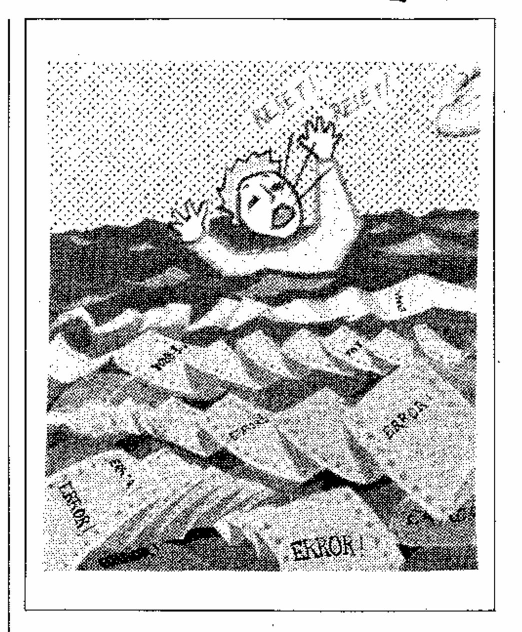
 ...
 ...          (Testteil der Schleife z. B.
              Vergleich)

              (Flag wird auf Stack gelegt)
UNTIL         (Ist Flag=0 erfolgt Sprung
              nach Begin)

 ...          (Flag ist ungleich 0; Pro-
              gramm wird fortgesetzt)
 ...
```

Das Beispiel mit den Zahlen:

```
: ZAHLEN
    0
    BEGIN
      1+ (Zahl um 1 erhöhen)
```




```
      DUP .      (Zahl ausgeben)
      DUP 10 =   (Ist 10 erreicht?)
    UNTIL
;
```

ZAHLEN <RETURN> 1 2 3 4 5 6 7 8 9 10 OK

In dem obigen Beispiel wird die Schleife so lange wiederholt, bis die auf dem Stack liegende Zahl den Wert 10 erreicht hat.

Eine weitere Anwendung wäre beispielsweise das Abfragen der Tastatur:

```
: INKEY (Auf <RETURN> wird gewartet)
    BEGIN
      KEY
      13 = (Ist <RETURN> gedrückt)
    UNTIL
    ." RETURN WURDE GEDRUECKT"
;
```

INKEY <RETURN>

<RETURN> RETURN WURDE GEDRUECKT OK

Zum Schluß das Beispiel noch einmal in Basic:

10 GET E$: IF E$="" THEN 10

20 IF ASC(E$(<> 13 THEN 10

30 PRINT „RETURN WURDE GEDRUECKT"

Forth-Kurs

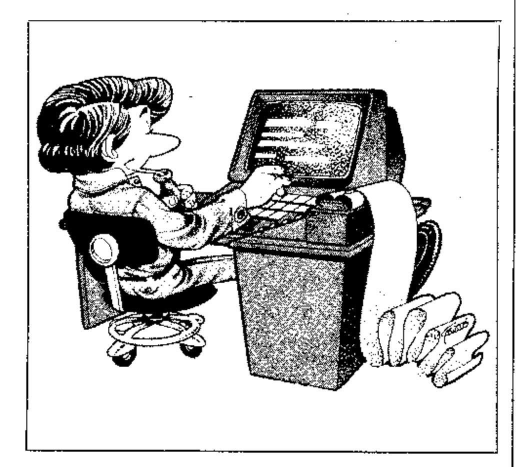
Die Endlosschleife Begin ... Again

Diese Programmschleife kennt keine Aussprungbedingung, ist also endlos. Die Hauptanwendung dieses Befehlswortes liegt im Aufruf bzw. in der unbegrenzt häufigen Abarbeitung eines (periodisch zu durchlaufenden) Hauptprogrammes. Die Syntax dieser Schleife entspricht weitgehend der BEGIN-UNTIL-Schleife.

Beispiel:

```
: TEST
    BEGIN
    . "COMPUTRONIC" CR (diese Anweisung wird endlos ausgeführt)
    AGAIN
;
```

Zu bemerken ist noch, daß es sich bei dieser Schleifenart um die schnellste *FORTH-Schleife* handelt.



BEGIN-WHILE-Schleife

Auch diese Schleifenstruktur wird beliebig lange durchlaufen. Im Gegensatz zur BEGIN-UNTIL-Schleife erfolgt die Abfrage einer Aussprungbedingung (Flag) am Anfang des Schleifenkernes. Ist die Bedingung am Schleifenbegin nicht erfüllt, so wird die Schleife nicht ein einziges Mal durchlaufen. Das folgende Beispiel zeigt die Syntax dieser Schleife:

```
BEGIN
...        (Bedingung zur Erzeugung
            des Flags)
WHILE      (solange das Flag = 1 ist,
...         wird der folgende Teil
            durchlaufen)
REPEAT     (Rücksprung nach BEGIN)
```

In dem folgenden Beispiel wird die Tastatur abgefragt. Ist die Taste „C" gedrückt, so wird der Text „TASTE C IST GEDRÜCKT" ausgegeben. Die Schleife wird erst verlassen wenn eine andere Taste gedrückt wird.

```
: TASTE
   BEGIN
     KEY 67 - 0 = (Ist Taste „C" gedrückt?)
       WHILE
       . "TASTE C IST GEDRÜCKT"
CR
       REPEAT (Springe zurück)
;
```

Wir haben nun alle verfügbaren *FORTH-Schleifen* kennengelernt. Es gibt in *FORTH* auch die Möglichkeit noch weitere Arten von Schleifen zu definieren, doch dies ist dem etwas erfahreren Programmierern vorbehalten. Wer sich genauer damit beschäftigen möchte, muß auf andere Literatur zurückgreifen, da die Beschreibung dieser Definitionen den Rahmen dieses Kurses sprengen würde.

FORTH-Programmbeispiele:

Berechnung des größten gemeinsamen Teilers zweier Zahlen:

```
: TEILER (Größter gemeinsamer Teiler errechnen)
    BEGIN
      SWAP OVER MOD DUP
      0 =
    UNTIL
    DROP
;
27 21 TEILER ‹RETURN› 3 OK
 8  6 TEILER ‹RETURN› 2 OK
```

Die Berechnung wurde mit Hilfe des Euklidschen Algorithmus verwendet. Zuerst bestimmt man den Rest ›R‹ zweier Zahlen, ›A‹ und ›B‹, durch (A MOD B). Ist R=0, dann ist B der gesuchte Teiler. Ist R‹›0, so wird A=B und B=R. Danach wir die obige Division wieder ausgeführt. Solange, bis die Bedingung R=0 erfüllt ist.

Variablen und Konstanten in Forth

Obwohl die meisten Zahlenwerte und Rechenoperationen über den Stack geführt werden, kann man trotzdem auch Konstante und Varibalen definieren.
In Basic wird eine Konstante wie folgt definiert:

10 LET Y = 10

Der Konstanten Y wird der Zahlenwert 10 zugeordnet. In *FORTH* würde das gleiche so geschrieben:

10 CONSTANT Y ‹RETURN› OK

Durch diese Befehlsfolge wird im Wörterbuch eine Konstante „Y" eingetragen, die den Zahlenwert 10 enthält. Sobald der Variablennamen aufgerufen wird, erhält der Stack den entsprechenden Zahlenwert.

Beispiele:

```
12 Y + .  ‹RETURN›  22 OK
 5 Y ★ .  ‹RETURN›  50 OK
 Y Y ★ .  ‹RETRUN›  100 OK
```

Das ganze kann auch in einer Definition stehen:

```
: Test
    Y ★ .
;
```

7 TEST ‹RETURN› 70 OK

Wie wir wissen, können wir statt der Konstanten auch einfach folgendes schreiben:

: Y 10 ;

Dies hat zwar die gleiche Wirkung wie die Definition einer Konstanten, ist jedoch in der Ausführungszeit wesentlich länger.
Im Gegensatz zu den meisten BASIC-Versionen können die Namen der Variablen, Konstanten und Befehlen in den meisten *FORTH-Versionen* bis zu 31 Zeichen lang sein. Es ist so möglich, den Variablen einen sinnvollen Namen zu geben, um somit das Programm übersichtlicher zu machen.

Ähnlich wie die Konstanten werden in *FORTH* auch die Variablen definiert:

10 VARIABLE TAG

In diesem Beispiel wurde der Variablen „TAG" der Wert 10 zugewiesen. Der Unterschied zwischen Variablen und Konstanten besteht darin, daß die Konstante nach der Definition ihren Wert immer behält. Dieser wird beim Aufruf der Konstanten auf den Stack gelegt. „Beim Aufruf einer Variablen wird dagegen nicht der Wert der Variablen, sondern die Adresse dieser Variable auf den Stack gelegt. Zum Ändern bzw. Auslesen des Wertes einer Variablen, benötigen wir noch zwei weitere Befehle, nämlich den Befehl :Wert holen (Load) ; und den Befehl :Wert wegspeichern (STORE).

Für den STORE-Befehl wird in *FORTH* das „!" Zeichen verwendet. Dieser Befehl entspricht weitgehend dem bekannten POKE aus dem BASIC. Jedoch wird hier keine 8-Bit-Zahl, sondern eine 16-Bit-Zahl abgespeichert.
Die Syntax des STORE-Befehl:

! (n1 n2 adr —› n1)

Für den LOAD-Befehl wird in Forth das " " Zeichen verwendet. Dieser Befehl ist mit dem Peek-Befehl im Basic zu vergleichen. Auch hier wird eine 16-Bit-Zahl ausgelesen:

Die Syntax des LOAD-Befehl:

@ (n1 n2 adr —› n1)

Die eben vorgestellten Befehle LOAD und STORE arbeiten nur mit 16-Bit-Zahlen. Will man dagegen nur ein einziges Byte, also eine Dezimalzahl zwi-



schen 0 und 255 abspeichern, so verwendet man das Wort C! „Mit Ce wird ein Byte auf den Stapel geholt. Dabei werden die oberen 8 Bits zu Null gesetzt.

Wenn wir nur den Variablennamen, gefolgt von einem Dezimalpunkt (Print) eingeben, so erhalten wir die Adresse der Variablen. Diese kann je nach *FORTH-Version* und Anzahl der Definitionen verschieden sein.

TAG . ‹RETURN› 4104 OK
Möchte man den Wert der Variablen TAG auslesen, so geschied dies wie folgt:

TAG @ . ‹RETURN› 10 OK

Das Ändern einer Variablen geht ebenso einfach:

13 TAG ! ‹RETURN› OK
(Der Wert 13 wird gespeichert)

TAG @ . ‹RETURN› 13 OK
(Der Wert wird ausgelesen)

Auf ähnliche Weise kann man auch den Wert einer schon vereinbarten Konstanten ändern. Die Adresse einer Konstanten erhält man mit dem Befehlswort "'" (Einzelapostroph).
Die Syntax:
' (Name) (n1 n2 --› n1 n2 adr)
Die Adresse der Konstanten (Name) wird auf den Stack gebracht.

Beispiele:

20 CONSTANT TRONIC ‹RETURN› OK
(der Konstanten TRONIC wird der Wert 20 zugewiesen)

TRONIC . ‹RETURN› 20 OK
(der Wert wird ausgelesen)

30 ' TRONIC ! ‹RETURN› OK
(der Wert 30 wird in die Konstante geschrieben)

TRONIC . ‹RETURN› 30 OK
(nochmal überprüfen)

FORTH-Zahlensystem

Bei der Initialisierung von *Forth*, wird immer das Rechnen zur Basis 10, also die gewohnte Dezimalrechnung angenommen. *Forth* kann jedoch auch mit allen anderen Zahlensystemen rechnen. Will man z. B. mit Hexadezimalzahlen arbeiten, so kann mit dem Befehl „HEX" auf diesen Modus umgeschaltet werden. Danach erfolgen alle Ein- und Ausgaben in hexadezimaler Schreibweise.

HEX (n1 n2 n3 --› n1 n2 n3)

Möchte man wieder in den Dezimal-Modus zurück, so geschieht dies mit dem Befehl "DECIMAL".

DECIMAL (n1 n2 n3 --› n1 n2 n3)

Beispiel:

DECIMAL 90 HEX . ‹RETURN› 5A OK
Die Dezimalzahl 90 entspricht der Hexadezimalzahl 5A.

HEX 5A DECIMAL . ‹RETURN› 100 OK
Das umgekehrte Beispiel.

Möchte man ein anderes Zahlensystem verwenden, muß man in die System-Variable BASE den entsprechenden Wert speichern. Durch den Befehl HEX wird eine 16 und durch DECIMAL eine 10 in die Variable geschrieben.

Möchten wir beispielsweise das Binärsystem programmieren, so müssen wir nur die Zahl 2 (Zustand 0 und 1) in die Variable BASE bringen. Den Umschalt-Befehl nennen wir BIN.

: BIN 2 BASE ! ;

Mit dem Befehl BIN können wir nun jederzeit auf Binär-Modus umschalten.

```
DECIMAL 255 BIN .
        ‹RETURN› 11111111 OK
DECIMAL  16 BIN .
        ‹RETURN›    10000 OK
BIN 01010101 DECIMAL .
        ‹RETURN›       85 OK
BIN 00010000 DECIMAL .
        ‹RETURN›       16 OK
```

Hier ein kleines FORTH-Programm

Berechnung der Primzahlen zwischen zwei Grenzen.

```
4 VARIABLE ZEILE
: TEST      MOD 0=;
: DRUCKE DUP 4 .R ZEILE e DUP 0=
            IF CR DROP 4
            ELSE 1 -
            THEN
            DROP ;

: PTEST     DUP 2 / 2 DO DUP I TEST
            IF 0 LEAVE
            THEN LOOP DUP IF
            DRUCKE
            ELSE DROP
            THEN
            DROP ;

: PRIM      (ende anf --› )
CR 4        ZEILE !
DO I        PTEST LOOP CR ;

200 1 PRIM ‹RETURN›
```



```
  1    2    5    7   11
 13   17   19   23   29
 31   37   41   43   47
 53   59   61   67   71
 73   79   83   89   97
101  103  107  109  113
127  131  137  139  149
151  157  163  167  173
179  181  191  193  197
199   OK
```

Die Spezial-Schleifen

Die Programmiersprache *FORTH* enthält natürlich eine beachtliche Anzahl von Wörtern, die von den im letzten Teil vorgestellten Strukturelementen selber Gebrauch machen. Etliche von ihnen haben auch unter reinen Anwender-Gesichtspunkten große praktische Bedeutung.
Daneben existieren einige Worte, die im wesentlichen spezialisierte Maschinensprache-Schleifen aufrufen. Die typischen Vertreter dieser Gruppe sind:

CMOVE (adr1 adr2 n --› empty)
Die durch ›n‹ bestimmte Anzahl Bytes, werden von Speicherbereich adr1 nach Speicherbereich adr2 kopiert. Dabei wird so vorgegangen, daß zuerst die Bytes mit den niedrigsten Adressen, fortschreitend zu höheren Adressen, transportiert werden.

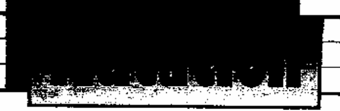
FILL (adr n1 n2 --› empty)
Der Speicherbereich ab Adresse adr wird mit n1 Bytes des Bitmusters n2 gefüllt.

ERASE (adr n --› empty)
Der Speicherbereich ab adr n wird mit n Bytes des Bitmusters 0 gefüllt. Also gelöscht!

BLANKS (adr n --› empty)
Die Speicherbereich ab adr n wird mit n Bytes des Bitmusters 32 (SPACE) gefüllt.

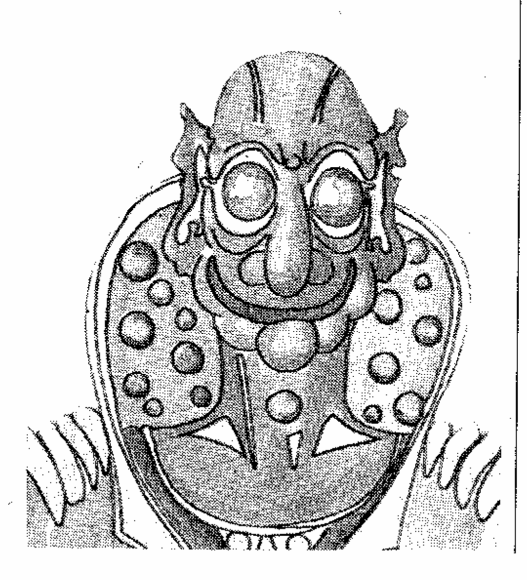
Diese Worte gehören naturgemäß zu den „schnellsten" *FORTH-Worten* überhaupt, da diese in Maschinensprache definiert sind.

weiter auf Seite 64





Eine Spielhallenversion, geschrieben auf dem Atari 800 XL, wo Reaktion und Geschwindigkeit gefragt sind.



Der wahnsinnige Dr. Zyclop hat auf dem Saturnmond Titan grauenhafte Mutanten gezüchtet, mit deren Hilfe er die Erde übernehmen will. In Schüben wachsender Stärke bewegen sich die Feinde auf Ihren Heimatplaneten zu. Die letzte Hoffnung sind 3 Raumschiffe der STARDUST-Klasse, mit denen Sie nun aufbrechen, um den Angriff zu stoppen. Sie können die Mutanten nur aufhalten, wenn Sie ihnen innerhalb kürzester Zeit große Verluste zufügen. Werden Sie die Versklavung der Menschheit verhindern können?

Zum Programm:
Die DATA-Zeilen 2010 bis 2100 müssen mit besonderer Sorgfalt eingegeben werden, da ein Fehler zum Absturz des Rechners führen könnte.

Die Maschinensprache-Unterroutine, die in B$ gespeichert ist (entsprechende Daten in Zeile 120), kann zum schnellen Löschen der Player-Missile-Grafik verwendet werden. Der Aufruf erfolgt durch X = USR (ADR [B$], Startadresse, Anzahl der Bytes). Die Startadresse gibt an, wo der Player im Speicher liegt, die Anzahl der Bytes bestimmt die Menge der zu löschenden Speicherzellen. Diese Zahl darf 255 nicht übersteigen.

Wer Zeit und Speicher sparen will, läßt einfach die Zeilen 10 000 bis 11 040 weg und ändert folgende Zeile:
180 READ B: IF B = -1 THEN 1000
Sollte vor dem Laden des Programms schon ein anderes gespielt worden sein, so sollte vor dem Start mit RUN der Befehl POKE 204,0 eingegeben werden, da ansonsten die Zeichen nicht definiert werden können.

Wer eine Pause einlegen möchte, drückt einfach die Tasten CONTROL und 1. Bei einem nochmaligen Drücken geht das Spiel weiter.

```
0 REM ********************
1 REM *                  *
2 REM * (c) 12.8.1984 by *
3 REM *                  *
4 REM *    Jens Berke    *
5 REM *                  *
6 REM ********************
7 REM
100 CLR :DIM A$(44),B$(18),SO(3):CH1=(PEEK(106)-8)*256
110 FOR A=1 TO 18:READ B:B$(A)=CHR$(B):NEXT A
120 DATA 104,104,133,207,104,133,206,104,104,168,169,0,145,206,136,208,251,96
130 IF PEEK(204)<>0 THEN 1000
140 FOR A=1 TO 44:READ B:A$(A)=CHR$(B):NEXT A
150 DATA 104,201,3,208,254,162,6,104,149,211,202,208,250,230,212,230,213,198,212
,208,4,198
160 DATA 213,240,18,161,216,129,214,230,216,208,2,230,217,230,214,208,234,230,21
5,208,230,96
170 Q=USR(ADR(A$),57344,CH1,1024):RESTORE 32000
180 READ B:IF B=-1 THEN 10000
190 FOR A=0 TO 7:READ C:POKE CH1+B*8+A,C:NEXT A:GOTO 180
500 POSITION 9,0:? SC:POSITION 24,0:? TR;" ":POSITION 36,0:? 60-TI;" ":RETURN
600 FOR A=10 TO 0 STEP -0.2:SOUND 0,255,10,A:NEXT A:RETURN
700 FOR B=1 TO 24:POSITION 0,A:? CHR$(156):NEXT B:RETURN
800 POKE 53248,A:POKE 53249,A:POKE 53250,A:RETURN
900 FOR A=1 TO 500:NEXT A:RETURN
1000 GRAPHICS 0:POKE 82,0:POKE 752,1:POKE 712,0:POKE 708,184:POKE 709,252:POKE
10,148:? CHR$(125)
1010 DL=PEEK(560)+256*PEEK(561):FOR A=6 TO 28:POKE DL+A,4:NEXT A:RESTORE 1090
1020 POKE 756,CH1/256:POSITION 1,0:? "SCORE : 0     / HITS : 0  / TIME : 60"
1030 Y=102:LE=3:SC=0:SG=0:TR=0
1040 PM=(PEEK(106)-12)*256:POKE 54279,PM/256:POKE 559,46:POKE 53277,3
1050 POKE 704,4:POKE 705,8:POKE 706,14:POKE 707,14:POKE 53278,0:POKE 53248,124:P
OKE 53249,124:POKE 53250,124
```



```
1060 FOR A=0 TO 3:POKE 53256+A,0:NEXT A
1070 Q=USR(ADR(B$),PM+512,255):Q=USR(ADR(B$),PM+768,255)
1080 FOR A=4 TO 8:POKE PM+512+Y+A,16:NEXT A
1090 FOR A=3 TO 9:POKE PM+768+Y+A,130:NEXT A
1100 FOR A=0 TO 2:READ B:POKE PM+768+Y+A,B:NEXT A:DATA 16,56,16
1110 FOR A=5 TO 7:READ B:POKE PM+640+Y+A,B:NEXT A:DATA 68,108,40
1120 POKE PM+640+Y+3,16
2000 FOR A=1536 TO 1771:READ B:POKE A,B:NEXT A
2010 DATA 120,72,138,72,152,72,173,120,2,201,11,240,7,201,7,240,16,76,54,6,198,2
03,165,203
2020 DATA 201,45,208,15,169,46,76,43,6,230,203,165,203,201,203,208,2,169,202,141
,0,208,141,1,208
2030 DATA 141,2,208,133,203,173,6,208,240,16,169,64,141,0,6,169,0,141,1,210,141,
3,208,76,78,6
2040 DATA 32,134,6,104,168,104,170,104,88,64,104,24,160,0,165,88,105,40,133,206,
165,89,133,207
2050 DATA 173,10,210,105,40,144,252,24,105,104,133,206,198,204,165,204,201,1,208
,4,169,4
2060 DATA 133,204,145,206,169,1,133,84,169,0,133,85,96,165,205,208,25,173,132,2,
208,89,141,234,6,169,1
2070 DATA 133,205,160,100,140,233,6,165,203,141,3,208,141,235,6,172,233,6,169,0,
145,208,152,56,233,2,24,201
2080 DATA 18,208,10,169,0,145,208,133,205,141,1,210,96,168,169,16,145,208,173,7,
208,240,15,169,96
2090 DATA 141,134,6,169,0,145,208,141,1,210,76,232,6,140,233,6,172,234,6,200,149
,0,210,140,234,6
2100 DATA 160,42,140,1,210,96,0,0,0
2110 A=INT((PM+896)/256):B=(PM+896)-A*256:POKE 208,B:POKE 209,A
2120 POKE 203,124:POKE 512,0:POKE 513,6:POKE 54286,192:POKE 204,4:POKE 205,0:POK
E DL,PEEK(DL)+128
2130 SOUND 1,0,0,2:SOUND 2,100,8,3
2140 POKE 18,0:POKE 19,0:POKE 20,0
3000 IF PEEK(1536)=64 THEN 4000
3010 IF PEEK(1670)=96 THEN 5000
3020 Q=USR(1621)
3030 A=INT((PEEK(20)+PEEK(19)*256)/50):TI=A-INT(A/60):IF 60-TI=0 THEN IF TR<SG*5
+20 THEN 9000
3040 ? CHR$(157):GOSUB 500
3050 FOR A=1 TO 50:NEXT A
3060 GOTO 3000
4000 RESTORE 4500:A=0:GOSUB 800:POKE 53251,PEEK(203)
4010 FOR A=0 TO 9:READ B:POKE PM+998+A,B:NEXT A
4020 FOR A=0 TO 200:SOUND 1,A,0,15:NEXT A:SOUND 1,0,0,0:SOUND 2,0,0,0
4030 A=1:GOSUB 700
4040 Q=USR(ADR(B$),PM+896,128)
4050 LE=LE-1:POKE DL+16,2
4060 IF LE=0 THEN 8000
4070 POSITION 10,11:? "RAUMSCHIFFE : ";:FOR A=1 TO LE:? CHR$(33);:NEXT A:GOSUB 6
00
4080 TI=0:TR=0:POKE 53278,0:POKE 205,0:POKE 203,124
4090 GOSUB 900:POSITION 8,11:? "ACHTUNG, ES GEHT WEITER.":GOSUB 600
4100 GOSUB 900:POKE DL+16,4:POSITION 0,11:? CHR$(156)
4110 A=124:GOSUB 800
4120 GOSUB 500:POKE 1536,120:POKE 1670,165:GOTO 2130
4500 DATA 0,0,161,66,17,68,146,32,0,0
5000 X1=INT((PEEK(1771)-45)/4):Y1=INT((PEEK(1769)-16)/4):TRAP 5050
5010 LOCATE X1,Y1,Z:IF Z=32 THEN Y1=Y1+1:GOTO 5010
5020 SC=SC+20:COLOR 37:PLOT X1,Y1
5030 FOR A=0 TO 100 STEP 4:SOUND 0,A,0,10:NEXT A:SOUND 0,0,0,0
5040 COLOR 32:PLOT X1,Y1:TR=TR+1:GOSUB 500:IF TR=SG*5+20 THEN 6000
5050 TRAP 40000:POKE 205,0:POKE 53278,0:POKE 1670,165:POKE 1771,PEEK(203):GOTO 3
000
```



```
6000 A=1:GOSUB 700:POKE 1536,64:POKE DL+16,2:POKE DL+17,2:SOUND 1,0,0,0:SOUND 2,
0,0,0
6010 FOR A=100 TO 50 STEP -5:FOR B=10 TO 0 STEP -2:SOUND 0,A,10,B:NEXT B:NEXT A:
SOUND 0,0,0,0
6020 POSITION 0,11:SG=SG+1:IF SG=7 THEN 7000
6030 ? " DU HAST AUCH DIESE WELLE UEBERSTANDEN   UND MUSST NUN ";SG*5+20;" MUTAN
TEN VERNICHTEN."
6040 GOSUB 600:TR=0:TI=0:POKE 205,0:Q=USR(ADR(B$),PM+896,128):POKE 53278,0:A=124
:GOSUB 800:POKE 203,124
6050 GOSUB 900:GOSUB 900:GOSUB 900:A=11:GOSUB 700:POKE DL+17,4:POSITION 4,11:? "
AUF ZUR NAECHSTEN MUTANTENWELLE."
6060 GOSUB 600:GOSUB 900:A=11:GOSUB 700:POKE DL+16,4:POKE 1536,120:POKE 1670,165
:GOTO 2130
7000 POKE 756,224:POSITION 8,11:? "DU HAST ES GESCHAFFT !!!":POSITION 5,12:? "DI
E MUTANTEN HABEN AUFGEGEBEN."
7010 GOSUB 600:POKE DL+18,2:GOTO 8030
8000 RESTORE 8010:FOR A=1 TO 3:READ B:SOUND 0,B,10,10:FOR C=1 TO 90:NEXT C:NEXT
A:SOUND 0,0,0,0
8010 DATA 114,102,128
8020 POSITION 10,11:? "G A M E    O V E R":POKE DL+18,2
8030 POSITION 3,13:? "DRUECKE START FUER EIN NEUES SPIEL"
8040 IF PEEK(53279)<>6 THEN 8040
8050 A=0:GOSUB 800:POKE 53251,0:RUN
9000 GOSUB 500:A=1:GOSUB 700:POKE DL+16,2:POKE DL+17,2:SOUND 1,0,0,0:SOUND 2,0,0
,0:POSITION 0,11
9010 ? " DIE ZEIT IST LEIDER ABGELAUFEN. DIE MU- TANTEN FALLEN NUN UEBER DIE ERD
E HER."
9020 GOSUB 600:GOSUB 900:GOSUB 900:GOSUB 900:A=11:GOSUB 700:POKE DL+17,4:GOTO 80
20
10000 GRAPHICS 17:POKE 709,12:POSITION 6,11:? #6;"mutation":POKE 712,12:POKE 708
,6:RESTORE 10190
10010 DL=PEEK(560)+256*PEEK(561)
10020 SO(0)=97:SO(1)=85:SO(2)=112:SO(3)=97:S=-1
10030 POKE 559,46:POKE 53277,3:POKE 53256,3:POKE 53257,3:PM=(PEEK(106)-12)*256:P
OKE 54279,PM/256:POKE 623,0
10035 Q=USR(ADR(B$),PM+512,255)
10040 FOR A=59 TO 64:POKE PM+512+A,255:POKE PM+640+A,255:NEXT A:POKE 704,0:POKE
705,0
10050 FOR A=0 TO 128 STEP 0.5:POKE 53248,A:POKE 53249,200-A+24
10060 SOUND 0,A+20,10,5:SOUND 1,A+21,10,5:SOUND 2,A+22,10,5:SOUND 3,A+23,10,5:NE
XT A
10070 SOUND 2,0,0,0:SOUND 3,0,0,0
10080 FOR A=0 TO 252 STEP 4:SOUND 0,A,0,10:SOUND 1,A+1,0,10:B=INT(RND(0)*7+1)*16
10090 POKE DL,B:NEXT A:SOUND 0,0,0,0:SOUND 1,0,0,0:POKE DL,112
10100 FOR A=1 TO 300:NEXT A
10110 POSITION 9,14:? #6;"BY":POSITION 5,16:? #6;"JENS BERKE":POSITION 3,20:? #6
;"druecke  start"
10115 IF PEEK(53279)=6 THEN 11000
10120 Z= NOT Z:POKE 623,Z:C=C+0.5:IF C>15 THEN C=0
10130 SETCOLOR 1,C,6:SETCOLOR 0,(15-C)*(Z=1),6+6*(Z=0)
10140 S=S+1:IF S=4 THEN S=0
10150 FOR A1=1 TO 10
10160 READ X:FOR A=15 TO 0 STEP -(X/10):SOUND 0,SO(S),12,A:NEXT A:SOUND 0,0,0,0
10170 NEXT A1:RESTORE 10190
10180 GOTO 10115
10190 DATA 10,22,10,22,10,10,10,22,22,10
11000 A=0:GOSUB 800:? #6;CHR$(125):POKE 708,212:POSITION 0,3:? #6;"DEINE AUFGABE
:"
11010 ? #6:? #6;"VERNICHTE  EINE  AN-"
11020 ? #6;"ZAHL  VON   MUTANTEN":? #6;"IN 60 SEKUNDEN.":? #6
```




```
11030 ? #6;"ZUERST  SIND  ES 20.":? #6;"SIE  NEHMEN DANN VON":? #6;"WELLE  ZU WE
LLE UM 5"
11040 ? #6;"ZU.":FOR A=1 TO 2000:NEXT A:GOTO 1000
32000 DATA 1,16,56,16,214,254,254,214,198
32010 DATA 2,0,168,136,136,168,16,16,68
32020 DATA 3,0,60,195,195,255,60,170,130
32030 DATA 4,0,4,17,17,21,21,34,136
32040 DATA 5,34,136,0,34,128,8,128,8,-1
```

Ski

Ski ist ein Spiel für eine Person, mit Joystick (Port 1), geschrieben in Atari-Basic.

Das Spiel ist auf den Atari-Computern 600 und 800 lauffähig.

Bei drücken der Starttaste erscheint die Skipiste. Vom Ziel bis zum Startplatz zeigt der Computer das Gelände bzw. die Piste, die zu bewältigen ist. Auf dem Bildschirm erscheinen neben der Piste Hindernisse (Hügel, Bäume) in den Farben Braun und Grün, die nicht berührt werden dürfen. Rote Fähnchen, ähnlich den Slalomstangen im Skisport, müssen durchfahren werden und dürfen ebenfalls nicht berührt werden. Das korrekte Durchqueren der Stangen wird mit 100 Punkten belohnt. Wichtig ist nun, in der vorgegebenen Zeit das Ziel, möglichst ohne ein Hindernis oder eine Fahnenstange zu streifen, zu erreichen. Gewonnen hat der Spieler mit der höchsten Punktzahl bzw. der schnellsten Zeit. Das Spiel beginnt bei drücken der roten Taste am Joystick. Drückt man den Joystick nach vorn, wird der Skifahrer schneller. Bei ziehen des Joysticks nach hinten, wird der Skifahrer langsamer. Neues Spiel: „Start" drücken!

```
100 DIM SCREEN$(1),PM$(1)
110 GOTO 230
120 REM + SKI +LINE
130 HI=INT(A/256)
140 LO=A-HI*256
150 RETURN
160 POKE 66,1
170 FOR W=1 TO 10
180 POKE 53279,0
190 POKE 53279,8
200 NEXT W
210 POKE 66,0
220 RETURN
230 POKE 12,119
240 POKE 13,228
250 GOSUB 1360
260 REM INITIALISIERUNG
270 REM SPIELER ROUTINE
280 POKE 559,62
290 POKE 54279,PMBASE
300 POKE 53277,3
310 POKE 704,0
320 P0=1024
330 YP=180
340 XP=128
350 PM$(P0)=CHR$(0)
360 PM$(P0+254)=CHR$(0)
370 PM$(P0+1)=PM$(P0)
380 DIM LINKS$(20),MITTE$(20),RECHTS$(20)
385 DIM CURR$(20),CRASH$(20),LOESCH$(20),DIR(8)
390 LOESCH$=CHR$(0)
400 LOESCH$(20)=CHR$(0)
410 LOESCH$(2)=LOESCH$
420 LINKS$=LOESCH$
430 MITTE$=LOESCH$
440 RECHTS$=LOESCH$
450 CRASH$=LOESCH$
460 FOR I=0 TO 15
470 LINKS$(I+2,I+2)=CHR$(PEEK(CHSET+208+I))
480 MITTE$(I+2,I+2)=CHR$(PEEK(CHSET+224+I))
490 RECHTS$(I+2,I+2)=CHR$(PEEK(CHSET+104+I))
500 CRASH$(I+2,I+2)=CHR$(PEEK(CHSET+240+I))
510 NEXT I
520 DIR(0)=0
530 DIR(1)=20
540 DIR(2)=19
550 DIR(3)=21
560 DIR(4)=1
570 FOR I=0 TO 3
580 DIR(I+5)=-DIR(I)
590 NEXT I
600 CURR$=MITTE$
610 PM$(P0+YP,P0+YP+20)=CURR$
620 SCR=SCR+10-PEEK(0)
630 POKE 20,0
640 POSITION 2,0:? #6;SCR;"  ";
645 POSITION 16,0:? #6;":  ";UFA;
650 IF PEEK(1)=0 THEN 1310
660 ST=STICK(0)
670 LINKS= NOT PTRIG(1)
680 RECHTS= NOT PTRIG(0)
690 LR=LINKS+2*RECHTS
700 CURR$=MITTE$
710 POKE 53248,XP
720 IF LINKS THEN CURR$=LINKS$
730 IF LR<>OLR THEN SV=2:TI=5
740 IF RECHTS THEN CURR$=RECHTS$
750 IF LR<>OLR THEN SV=4:TI=5
760 IF TI>0 THEN TI=TI-1
770 SOUND 0,SV,0,TI
780 IF LR=0 THEN SOUND 0,0,0,0:TI=0
790 XP=XP+LINKS-RECHTS
800 OLR=LR
810 UP=(ST=14 OR ST=10 OR ST=6)
```

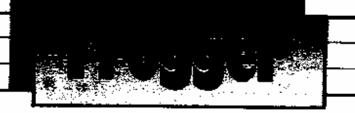


```
820 DOWN=(ST=5 OR ST=9 OR ST=13)
830 YP=YP-2*UP+2*DOWN
840 IF YP>200 THEN YP=200
850 IF YP<40 THEN YP=40
860 POKE 0,(YP-48)/48+1
870 IF PEEK(P0PF)=0 THEN 610
880 WHICH=INT(LOG(PEEK(P0PF))/LOG(2)+0.1)
890 POKE 0,0
900 PM$(P0+YP,P0+YP+20)=LOESCH$
910 POKE HITCLR,0
920 IF WHICH<>2 THEN 1130
930 REM POINTS
940 PTR=ASC(DLIST$(8))+256*ASC(DLIST$(9))
950 LINE=INT((YP-39)/8)+1
960 COL=INT((XP-49)/8)+1
970 LOC=PTR+LINE*20+COL
980 SOUND 0,0,0,0
990 FOR I=0 TO 8
1000 P=PEEK(LOC+DIR(I))
1010 IF P<128 OR P>192 THEN 1070
1020 POKE LOC+DIR(I),0
1030 SCR=SCR+(P=139)*50+(P=134)*100
1040 I=8
1050 NEXT I
1060 GOTO 1090
1070 NEXT I
1080 GOTO 610
1090 FOR W=15 TO 0 STEP -1
1100 SOUND 0,20,10,W
1110 NEXT W
1120 GOTO 610
1130 REM CRASH !
1140 SOUND 0,0,0,0
1150 PM$(P0+YP,P0+YP+20)=CRASH$
1160 POKE 704,56:FOR J=1 TO 5
1165 FOR K=3 TO 8
1170 SOUND 0,K,10,8:NEXT K:NEXT J
1180 FOR N=1 TO 25:NEXT N:POKE 704,14
1185 FOR L=1 TO 25:NEXT L
1190 PM$(P0+YP,P0+YP+20)=LOESCH$
1200 YP=200
1210 PM$(P0+YP,P0+YP+20)=CURR$
1220 POKE 0,1
1230 SOUND 0,0,0,0:POKE 704,0
1240 XP=INT(72+90*RND(0))
1250 IF PEEK(P0PF)<>0 THEN POKE HITCLR,0:
     GOTO 1240
1260 POKE 53248,XP
1270 POKE HITCLR,0:UFA=UFA+1
1280 IF UFA>10 THEN SCR=SCR-500
1285 SCR=SCR-50
1290 IF SCR<0 THEN SCR=0
1300 GOTO 610
1310 POSITION 7,0:? #6;"GAME OVER"
1320 SOUND 0,0,0,0
1330 SCREEN$(326,338)="Press@trigger"
1340 IF STRIG(0)<>0 THEN 1340
1350 GOTO 1410
1360 REM INITIALISIERUNG
1370 GRAPHICS 0:POKE 752,1:POKE 82,0
1375 SETCOLOR 2,0,0
1380 POSITION 0,22
1385 ? "Brauchst du eine Spielanleitung ?"
1390 OPEN #1,4,0,"K:":GET #1,A
1400 IF A=74 THEN CLOSE #1:GOSUB 3210
1410 GRAPHICS 17
1420 HILO=120:UFA=0
1430 POKE 53248,0
1440 POKE 0,0
1450 POKE 712,14
1460 POKE 709,198
1470 POKE 711,136
1480 POKE 710,250:POKE 708,85
1490 P0PF=53252
1500 HITCLR=53278
1510 POKE HITCLR,0
1520 SCRBASE=PEEK(106)-16
1530 PMBASE=SCRBASE-8
1540 CHBASE=PMBASE
1550 VVTP=PEEK(134)+256*PEEK(135)
1560 STARTP=PEEK(140)+256*PEEK(141)
1570 A=SCRBASE*256-STARTP
1580 GOSUB HILO
1590 POKE VVTP+2,LO
1600 POKE VVTP+3,HI
1610 POKE VVTP+4,1
1620 POKE VVTP+5,16
1630 POKE VVTP+6,1
1640 POKE VVTP+7,16
1650 A=PMBASE*256-STARTP
1660 GOSUB HILO
1670 POKE VVTP+10,LO
1680 POKE VVTP+11,HI
1690 POKE VVTP+12,1
1700 POKE VVTP+13,8
1710 POKE VVTP+14,1
1720 POKE VVTP+15,8
1730 CHSET=CHBASE*256
1740 IF PEEK(CHSET+9)<>6 THEN GOSUB 2640
1750 GOSUB 2950
1760 Z=USR(1638)
1770 POKE 756,CHBASE
1780 RESTORE 1840
1790 DIM T$(20),DLIST$(40),TOPLINE$(20)
1800 A=ADR(DLIST$)
1810 GOSUB HILO
1820 POKE 561,HI
1830 POKE 560,LO
1840 DATA 112,112,112,70,0,0,102,0,0
1850 FOR I=1 TO 9
1860 READ A
1870 DLIST$(I)=CHR$(A)
1880 NEXT I
1890 FOR I=1 TO 20
1900 DLIST$(I+9)=CHR$(6+32)
1910 NEXT I
1920 DLIST$(30)=CHR$(6)
1930 DLIST$(31)=CHR$(65)
1940 DLIST$(32)=CHR$(PEEK(560))
1950 DLIST$(33)=CHR$(PEEK(561))
1960 SCREEN$(1)=CHR$(0)
1970 SCREEN$(4095)=CHR$(0)
1980 SCREEN$(2)=SCREEN$
1990 TOPLINE$=SCREEN$
2000 A=ADR(TOPLINE$)
2010 GOSUB HILO
2020 DLIST$(5,5)=CHR$(LO)
2030 DLIST$(6,6)=CHR$(HI)
2040 POKE 88,LO:POKE 89,HI
2050 POSITION 8,0:? #6;"SkI !";
2055 REM LETZTE BUCHSTABE VON 'SKI'
2056 REM IN ZEILE 2050 INVERSE
2060 SCREEN$(409,414)="ziel"
2065 REM 'ZIEL' IN ZEILE 2060
2066 REM INVERSE SCHREIBEN
2070 A=SCRBASE*256
2080 FOR L=24 TO 198
2090 A=A+20
2100 GOSUB HILO
2110 T$=CHR$(LO)
2120 T$(2)=CHR$(HI)
2130 DLIST$(8,9)=T$
2140 S=L*20+1:E=S+19
2150 LFLEN=INT(7*RND(0)+1)
2160 RTLEN=INT(7*RND(0)+1)
2170 FOR I=1 TO LFLEN
2180 Z=INT(8*RND(0))
2190 T$(I)=CHR$((72+Z)*(Z<2))
2200 NEXT I
2210 SCREEN$(S,S+LFLEN)=T$(1,LFLEN)
2220 FOR I=1 TO RTLEN
2230 Z=INT(8*RND(0))
```




```
2240 T$(I)=CHR$((72+Z)*(Z<2))
2250 NEXT I
2260 SCREEN$(E-RTLEN,E)=T$(1,RTLEN)
2270 IF RND(1)>0.5 THEN 2420
2280 IF L-LAST<10 THEN 2420
2290 LAST=L
2300 SKEW=1:IF RND(0)>0.5 THEN SKEW=-1
2310 SP=INT(7*RND(0)+5)
2320 SCREEN$(S+SP,S+SP)=CHR$(134)
2330 FOR I=0 TO INT(RND(1)*2.76)
2340 RT=SP+I*40+SKEW*(I+1)
2350 LF=RT+20-SKEW*2
2360 SCREEN$(S+LF,S+LF)=CHR$(204)
2370 SCREEN$(S+RT,S+RT)=CHR$(204)
2380 RT=SP+I*40+SKEW*I
2390 SCREEN$(S+RT,S+RT)=CHR$(134)
2400 NEXT I
2410 GOTO 2540
2420 IF RND(1)>0.025 THEN 2460
2430 SP=S+INT(13*RND(0)+5)
2440 SCREEN$(SP,SP)=CHR$(7)
2450 GOTO 2540
2460 IF RND(1)>0.025 THEN 2500
2470 SP=S+INT(13*RND(0)+5)
2480 SCREEN$(SP,SP)=CHR$(10)
2490 GOTO 2540
2500 IF RND(1)>0.1 THEN 2540
2510 SP=S+INT(13*RND(0)+5)
2520 SCREEN$(SP,SP)=CHR$(139)
2530 GOTO 2540
2540 NEXT L
2550 A=A+200
2560 GOSUB HILO
2570 T$=CHR$(LO)
2580 T$(2)=CHR$(HI)
2590 DLIST$(8,9)=T$
2600 GOSUB 160
2610 IF STRIG(0) THEN 2600
2620 A=USR(1536,ADR(DLIST$(8)),176)
2630 RETURN
2640 FOR I=0 TO 7
2650 POKE CHSET+I,0
2660 NEXT I
2670 FOR I=128 TO 471
2680 POKE CHSET+I,PEEK(57344+I)
2690 NEXT I
2700 RESTORE 2780
2710 READ A
2720 IF A=-1 THEN RETURN
2730 FOR J=0 TO 7
2740 READ B
2750 POKE CHSET+A*8+J,B
2760 NEXT J
2770 GOTO 2710
2780 DATA 1,0,6,14,28,24,32,0,128
2790 DATA 6,192,192,220,20,28,7,5,7
2800 DATA 7,0,0,24,52,44,60,24,0
2810 DATA 8,16,56,56,124,124,254,16,16
2820 DATA 9,8,28,62,62,62,8,8,0
2830 DATA 10,0,56,94,106,94,116,56,0
2840 DATA 11,0,119,69,117,21,119,0,0
2850 DATA 12,8,24,56,120,8,8,8,8
2860 DATA 14,254,89,24,156,82,33,16,8
2870 DATA 13,0,0,0,48,88,56,16,186
2880 DATA 27,127,154,24,57,74,132,8,16
2890 DATA 26,0,0,0,12,26,28,8,93
2900 DATA 29,186,89,24,154,170,198,65,65
2910 DATA 28,0,0,24,60,60,24,24,60
2920 DATA 31,10,24,24,0,0,0,0,0
2930 DATA 30,1,18,36,74,161,18,156,77
2940 DATA -1
2950 RESTORE 3020
2960 FOR I=1536 TO 1648
2970 READ A
2980 POKE I,A
2990 NEXT I
3000 RETURN
3010 REM BITTE GENAU EINGEBEN
3020 DATA 169,0,133,0,169,1
3030 DATA 141,99,6,169,8,141
3040 DATA 98,6,104,104,133,7
3050 DATA 104,133,6,104,104,133
3060 DATA 1,162,6,160,35,169
3070 DATA 7,32,92,228,96,216
3080 DATA 165,0,240,55,165,1
3090 DATA 240,51,206,99,6,173
3100 DATA 99,6,208,43,165,0
3110 DATA 141,99,6,206,98,6
3120 DATA 174,98,6,142,5,212
3130 DATA 208,27,160,0,56,177
3140 DATA 6,233,20,145,6,160
3150 DATA 1,177,6,233,0,145
3160 DATA 6,169,7,141,98,6
3170 DATA 141,5,212,198,1,76
3180 DATA 98,228,0,0,0,0
3190 DATA 104,162,228,160,98
3200 DATA 169,7,32,92,228,96
3210 REM SPIELANLEITUNG
```

```
3220 GRAPHICS 0:POKE 752,1:SETCOLOR 2,0,0:POSITION 7,1:? " S K I "
3230 POSITION 1,4:? "SKI ist ein Spiel fuer eine Person mit"
3235 REM DAS WORT 'SKI' IN ZEILE 3220
3236 REM UND ZEILE 3230 INVERSE SCHREIBEN
3240 POSITION 1,5:? "Joystick (Port (1))."
3250 POSITION 1,6:? "Zu Beginn des Spieles wird durch "
3260 POSITION 1,7:? "scrollen des Bildschirmes von unten "
3270 POSITION 1,8:? "nach oben die gesamte Strecke fuer den"
3280 POSITION 1,9:? "Abfahrtslauf gezeigt."
3290 POSITION 1,10:? "Die Baeume sowie die Hindernisse als"
3300 POSITION 1,11:? "auch die Faehnchen duerfen nicht "
3310 POSITION 1,12:? "beruehrt werden, da bei Kollision 50 "
3320 POSITION 1,13:? "Punkte abgezogen werden."
3330 POSITION 1,14:? "Zwischen den Faehnchen und auf dem "
3340 POSITION 1,15:? "Gelaende liegenden Punkte bekommt man "
3350 POSITION 1,16:? "bei Ueberfahren."
3360 POSITION 1,17:? "Wenn die gesamte Strecke gezeigt wurde"
3370 POSITION 1,18:? "ertoent ein Schnarren vom Lautsprecher"
3380 POSITION 1,19:? "des Atari, und durch dreucken des "
3390 POSITION 1,20:? "Triggers (roter Knopf) wird das,sowie"
3400 POSITION 1,21:? "jedes weitere Spiel begonnen."
3410 POSITION 25,22:? "---> TRIGGER"
3420 IF STRIG(0)<>0 THEN 3420
3430 RETURN
```





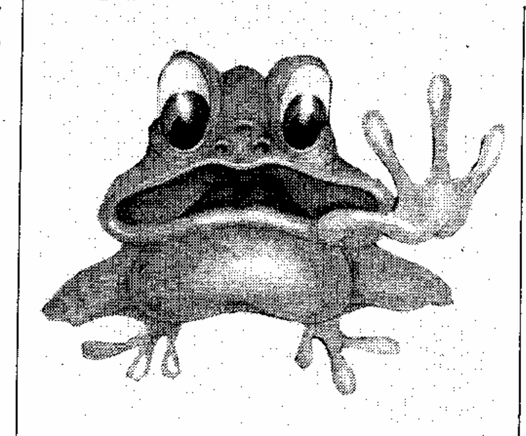
Topprogramm

Nachdem die Spielversion „Frogger" (für den Commodore 64) auf den Markt kam, wurde der kleine Frosch rasch zu einem Riesenerfolg!

Wir stellen Ihnen nun in unserer Ausgabe eine Spielversion für Ihren ZX-Spectrum vor.

Während des Spielverlaufs müssen Sie versuchen, alle fünf Frösche, die Ihnen zur Verfügung stehen, in ihren Unterschlupf zu bringen. Zunächst muß jeder Frosch eine vielbefahrene Straße überqueren. Auf dem Mittelweg angekommen, kann er sich erst einmal ausruhen und die Flußüberquerung vorbereiten. – Ab Level 3 wird die Ruhepause auf dem Mittelweg durch eine sich hin- und herbewegende Schlange erschwert. – Auf die andere Seite des Flußes kann der Frosch nur gelangen, wenn er sich abwärtstreibende Schildkröten und Baumstämme zu Hilfe nimmt. Auf ihnen kann er sich, ohne ins Wasser zu müssen, kurzfristig bewegen. Aber Vorsicht – die Schildkröten tauchen des öfteren unter. An dieser Stelle kann also ein Frosch verlorengehen. Das Untertauchen der Schildkröten kündigt sich durch das Wechseln der Farbe an.

Ist der Fluß erfolgreich überquert worden, müssen Sie ihn in seinen Unterschlupf einquartieren. Vergewissern Sie sich aber vorher, ob sich nicht schon eines dieser fürchterlichen Krokodile in der Behausung aufhält. Sind alle fünf Frösche gut untergebracht, erfahren Sie über den Score, wieviel Zeit Sie benötigt haben. – Gesteuert wird mit der Tastatur: ›2‹ oben, ›W‹ unten, ›9‹ links, ›0‹ rechts. Mit ›P‹ kann das Spiel vorübergehend gestoppt werden. Bei ›S‹ können Sie es



wieder starten. Eine Besonderheit: Sollte der eigentlich gute Sound störend wirken, kann dieser durch die Taste ›T‹ abgeschaltet werden. Das Spiel ist in sieben Levels unterteilt. Wir wünschen viel Vergnügen!

Programmeingabe

Geben Sie Programmteil >1< ein und starten Sie es mit RUN.
Das Programm überprüft nun, ob die Datazeilen richtig eingegeben wurden. Falls nicht, verbessern Sie diese.
Danach fragt das Programm nach den Zahlen, die Sie aus der Tabelle von links nach rechts eingeben. Ganz links steht immer die Adresse der ersten Zahl für diese Zeile. Diese Adresse dient nur zu Orientierung und braucht nicht mit eingegeben werden.
Nach jeweils 50 Eingaben überprüft das Programm, ob die 50 eingegebenen Zahlen mit der Prüfsumme übereinstimmen. Stimmt es nicht, müssen die vorigen 50 Zahlen nochmals eingegeben werden. Wenn alles richtig war, können Sie weiter eingeben oder das Eingegebene abspeichern oder Vorangegangenes laden.

Wenn Sie ihre eingegebenen Zahlen abspeichern wollen, müssen Sie später beim Laden zuerst CLEAR 50 000 eingeben und dann mit LOAD "" das Eingebeprogramm laden. Danach kommen Sie in das Menü. Wenn Sie jetzt „L" für Load engeben, werden die Zahlen geladen, die Sie vorher eingegeben haben. Nun können Sie dort weitermachen, wo Sie vorher aufgehört haben.

Wenn Sie alle Zahlen eingegeben haben, erscheint eine Fehlermeldung ‚Out of Data'. Löschen Sie nun den Computer durch NEW (die eingegebenen Zahlen werden dadurch nicht zerstört) und geben Sie nun Programm 2 ein.

Speichern Sie nun alles wie folgt ab:
SAVE „FROGGER" LINE 10: SAVE „FROGGER" CODE 60 000, 3860.
Nachdem Sie das Gespeicherte mit VERIFY"": VERIFY"" CODE überprüft haben, löschen Sie den Computer total (RANDOMISE USR 0 eingeben oder Stecker ziehen) und laden Sie das Programm mit LOAD"". Nun wird Programm 2 geladen und dieses lädt das Maschinenprogramm und startet es sofort. Sollte der Computer abstürzen oder das Spiel nicht anfangen, müssen Sie die Zahlen im Speicher mit den Zahlen in der Tabelle vergleichen und gegebenenfalls Verbesserungen ausführen. Da das Programm ihre Eingaben durch Prüfsummen intensiv überprüft, dürfte ein Fehler schon viel eher auftauchen.

```
10 REM ************************
20 REM *                      *
30 REM *     F R O G G E R    *
40 REM *                      *
50 REM *   COPYRIGHT 1984    *
60 REM *                      *
70 REM *  HANS PETER  FUNKE   *
80 REM *                      *
90 REM ************************
```

Ausgewählt von der Redaktion zum Topprogamm für den ZX-Spectrum 48K!

```
 100 REM
 110 REM PROGRAMM 1
 120 REM
 130 CLEAR 50000: POKE 23609,20
 140 POKE 23658,12
 150 LET PF=0
 160 FOR N=1 TO 78
 170 READ P
 180 LET PF=PF+P
 190 NEXT N
 200 IF PF<>443156 THEN  PRINT "FEHLER IN ZEILE 1000 ODER 1010.": BEEP 1,0: STOP

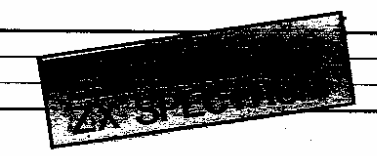
 210 RESTORE
 220 LET ADR=60000
 230 READ SUM
 240 LET PR=0
 250 FOR K=1 TO 50
 260 PRINT ADR;" - ";
 270 INPUT "WERT:";W
 280 IF W>255 THEN  GO TO 270
 290 PRINT W
 300 POKE ADR,ABS W
 310 LET PR=PR+W
 320 LET ADR=ADR+1
 330 NEXT K
 340 IF SUM<>PR THEN  PRINT  FLASH 1;" FEHLER!": BEEP 2,0: LET ADR=ADR-50: GO TO
 240
 350 INPUT "WEITER  LADEN  SAVEN  ? :";A$
 360 CLS
 370 IF A$="W" THEN  GO TO 230
 380 IF A$="L" THEN  PRINT "STARTE DEN RECORDER !.": LOAD "FROGGER"CODE : GO TO
230
 390 SAVE "FROGGER" LINE 350: SAVE "FROGGER"CODE 60000,ADR-60000
 400 PRINT "VERIFY."'"WENN EIN FEHLER AUFTRETEN SOLLTESTARTEN SIE DAS PROGRAMM M
IT    GOTO 250.": VERIFY "": VERIFY ""CODE : PRINT "O.K. !"
 420 DATA 4617,2876,2319,2509,4511,4206,3714,2507,2554,2309,2913,4877,6778,4681,
4930,5658,4397,4368,6637,8017,7258,6190,6905,7371,7751,5624,5446,5774,5038,5510,
5112,5835,6084,6773,6904,5405,6739,6610,4934
 430 DATA 5543,6340,6841,6278,3731,6927,5033,7025,6305,6930,6501,7207,7207,6271,
9179,6835,4812,4197,5801,8467,6823,4810,7240,8328,7477,5047,6159,5774,5583,5596,
6246,7135,6747,4279,5803,6032,6897,6133,976
```

```
60000 - 100 100 96 88 96 64 3 130 130 224 88 224 64 7
60014 - 110 110 1 191 89 191 72 13 150 150 0 224 89 224
60028 - 72 15 110 110 1 63 90 63 80 17 140 140 0 96
60042 - 90 96 80 19 150 6 150 1 167 88 12 160 88 192
60056 - 0 15 1 185 88 15 191 88 191 64 5 200 85 200
60070 - 1 34 89 15 32 89 64 0 20 1 52 89 15 63
60084 - 89 63 72 9 10 3 2 2 2 2 0 2 2 2
60098 - 1 16 200 0 0 0 0 0 0 50 1 0 0 232
60112 - 3 208 7 19 0 220 5 244 1 0 0 255 0 21
60126 - 16 0 0 63 0 0 5 0 0 0 22 0 3 83
60140 - 67 79 82 69 32 32 76 69 86 69 76 32 32 70
```




```
60154 - 82 79 71 83 32 32 72 73 71 72 58 22 0 0
60168 - 16 3 17 1 90 69 73 84 1 48 61 61 61 48
60182 - 61 61 61 46 46 48 48 54 54 54 54 46 46 48
60196 - 48 54 54 36 36 40 46 48 54 61 61 96 122 122
60210 - 122 96 122 122 122 92 92 96 96 108 108 108 108 92
60224 - 92 96 96 108 108 72 72 80 92 96 108 122 122 22
60238 - 2 6 139 131 32 139 134 32 137 134 32 137 134 32
60252 - 137 134 32 139 131 32 139 134 22 3 6 142 32 32
60266 - 142 137 32 138 133 32 138 32 32 138 32 32 142 32
60280 - 32 142 137 22 4 6 138 32 32 139 136 32 138 133
60294 - 32 138 135 32 138 135 32 138 32 32 139 136 22 5
60308 - 6 138 32 32 138 133 32 134 137 32 134 137 32 134
60322 - 137 32 142 140 32 138 133 22 7 0 67 79 80 89
60336 - 82 73 71 72 84 32 49 57 56 52 32 32 72 65
60350 - 78 83 32 80 69 84 69 82 32 70 85 78 75 69
60364 - 22 10 4 83 84 69 85 69 82 85 78 71 58 22
60378 - 12 8 50 22 14 8 60 22 16 4 57 32 109 32
60392 - 32 32 61 32 48 22 18 8 62 22 20 8 87 22
60406 - 11 16 80 32 58 32 80 65 85 83 69 22 13 16
60420 - 84 32 58 32 84 79 78 32 65 78 32 65 85 83
60434 - 22 15 16 83 32 58 32 83 84 65 82 84 22 18
60448 - 16 72 73 71 72 83 67 79 82 69 58 22 20 18
60462 - 17 5 41 0 196 3 55 0 38 3 65 0 122 2
60476 - 82 0 88 3 62 0 88 3 123 0 196 3 55 0
60490 - 62 4 49 0 38 3 65 0 122 2 82 0 84 2
60504 - 87 0 203 2 73 0 203 2 147 0 38 3 65 0
60518 - 6 24 205 68 14 17 232 234 1 29 0 205 60 32
60532 - 62 3 33 0 88 54 51 35 6 10 54 36 35 54
60546 - 36 35 197 6 5 54 51 35 16 251 125 254 36 32
60560 - 2 46 33 193 16 232 46 64 54 9 35 124 254 89
60574 - 32 248 125 254 96 32 243 205 177 236 17 32 1 25
60588 - 205 177 236 24 8 6 32 54 54 35 16 251 201 205
60602 - 128 243 205 142 237 6 24 205 106 237 237 75 229 234
60616 - 205 43 45 205 227 45 6 12 205 106 237 58 220 234
60630 - 60 6 0 79 205 43 45 205 227 45 205 229 236 24
60644 - 36 6 17 205 106 237 58 228 234 79 6 5 254 0
60658 - 40 15 254 5 56 2 14 5 62 63 215 13 40 6
60672 - 16 248 201 62 32 215 16 251 201 33 3 3 34 6
60686 - 235 58 184 234 111 205 120 237 33 7 7 34 6 235
60700 - 46 6 205 120 237 33 7 23 34 6 235 46 7 205
60714 - 120 237 62 5 50 9 235 33 5 7 34 6 235 205
60728 - 219 237 33 5 25 34 6 235 205 219 237 33 9 2
60742 - 34 6 235 205 219 237 33 9 20 34 6 235 205 219
60756 - 237 175 50 11 235 62 3 50 9 235 205 241 237 62
60770 - 1 50 11 235 205 136 247 201 62 51 50 143 92 62
60784 - 22 215 62 1 215 120 215 201 17 5 235 1 7 0
60798 - 205 60 32 62 33 215 69 62 34 215 16 251 62 35
60812 - 215 201 33 33 23 34 138 92 34 130 92 33 224 80
60826 - 34 134 92 62 253 205 1 22 1 4 0 17 12 235
60840 - 205 60 32 62 2 205 1 22 33 255 82 34 197 234
60854 - 33 229 82 6 27 229 14 4 54 255 36 13 32 250
60868 - 225 35 16 243 33 229 90 22 2 6 27 114 120 254
60882 - 22 32 2 22 4 35 16 245 201 17 5 235 1 7
60896 - 0 205 60 32 58 185 234 71 62 36 215 62 37 215
60910 - 16 248 201 17 5 235 1 7 0 205 60 32 201 237
60924 - 75 221 234 205 131 37 205 213 45 50 223 234 237 75
60938 - 221 234 205 56 37 205 241 43 50 224 234 62 22 215
60952 - 58 221 234 215 58 222 234 215 58 223 234 230 120 246
60966 - 4 50 143 92 58 225 234 215 201 62 22 215 58 221
60980 - 234 215 58 222 234 215 58 223 234 50 143 92 58 224
60994 - 234 215 201 1 254 247 58 191 234 87 237 120 230 2
61008 - 186 40 7 50 191 234 183 202 236 238 1 254 251 58
```

```
61022 - 192 234 87 237 120 230 2 186 40 6 50 192 234 183
61036 - 40 94 1 254 239 58 193 234 87 237 120 230 2 186
61050 - 40 7 50 193 234 183 202 12 239 1 254 239 58 194
61064 - 234 87 237 120 230 1 186 40 7 50 194 234 183 202
61078 - 38 239 1 254 223 237 120 203 71 40 9 1 50 0
61092 - 11 120 177 32 251 201 1 0 0 62 253 219 254 203
61106 - 79 32 5 62 2 211 254 201 16 241 13 32 238 237
61120 - 95 230 126 15 246 1 230 7 211 254 24 222 58 221
61134 - 234 254 21 200 205 47 238 62 2 50 231 234 58 221
61148 - 234 60 60 50 221 234 62 91 50 225 234 205 251 237
61162 - 24 82 58 221 234 254 1 200 205 47 238 62 1 50
61176 - 231 234 58 221 234 61 61 50 221 234 62 63 50 225
61190 - 234 205 251 237 24 50 58 222 234 254 0 200 205 47
61204 - 238 58 222 234 61 50 222 234 62 92 50 225 234 205
61218 - 251 237 24 24 58 222 234 254 31 200 205 47 238 58
61232 - 222 234 60 50 222 234 62 64 50 225 234 205 251 237
61246 - 58 204 234 183 192 62 2 17 128 128 1 5 0 211
61260 - 254 29 32 3 238 24 90 16 246 203 42 203 18 13
61274 - 81 32 238 201 221 126 0 61 221 119 0 183 192 58
61288 - 221 234 221 190 20 32 17 58 222 234 254 31 32 6
61302 - 62 1 50 199 234 201 60 50 222 234 221 126 1 61
61316 - 221 119 1 221 126 11 61 221 119 11 221 126 2 221
61330 - 119 0 221 126 1 183 40 56 221 126 3 254 1 32
61344 - 12 221 126 1 254 8 48 5 62 11 221 119 6 221
61358 - 110 4 221 102 5 58 185 234 203 39 71 221 126 6
61372 - 119 35 229 125 221 190 9 32 8 225 221 110 7 221
61386 - 102 8 229 225 16 233 24 48 237 95 246 20 230 31
61400 - 221 119 1 221 126 3 183 32 6 60 221 119 3 24
61414 - 9 175 221 119 3 62 10 221 119 1 62 13 221 119
61428 - 6 221 126 3 183 32 178 62 9 221 119 6 24 171
61442 - 221 126 11 183 40 55 221 126 12 183 40 12 221 126
61456 - 11 254 5 48 5 62 11 221 119 15 221 110 13 221
61470 - 102 14 58 185 234 203 39 71 221 126 15 119 35 229
61484 - 125 221 190 9 32 8 225 221 110 7 221 102 8 229
61498 - 225 16 233 24 48 237 95 246 12 230 31 221 119 11
61512 - 221 126 12 183 32 6 60 221 119 12 24 9 175 221
61526 - 119 12 62 8 221 119 11 62 13 221 119 15 221 126
61540 - 12 183 32 178 62 9 221 119 15 24 171 221 110 16
61554 - 221 102 17 221 94 18 221 86 19 205 177 240 221 110
61568 - 4 221 102 5 35 125 221 190 9 32 6 221 110 7
61582 - 221 102 8 221 117 4 221 116 5 221 110 13 221 102
61596 - 14 35 125 221 190 9 32 6 221 110 7 221 102 8
61610 - 221 117 13 221 116 14 201 229 213 225 43 6 8 229
61624 - 213 26 245 197 1 31 0 237 184 193 241 18 209 225
61638 - 36 20 16 237 209 213 225 43 26 245 1 31 0 237
61652 - 184 241 18 201 229 213 225 35 6 8 229 213 26 245
61666 - 197 1 31 0 237 176 193 241 18 209 225 36 20 16
61680 - 237 209 213 225 35 26 245 1 31 0 237 176 241 18
61694 - 201 58 220 234 254 2 216 58 218 234 183 32 4 205
61708 - 104 241 201 58 219 234 61 50 219 234 183 192 62 100
61722 - 50 219 234 62 22 215 62 11 215 58 217 234 215 62
61736 - 54 50 143 92 62 32 215 62 32 215 58 217 234 254
61750 - 30 32 13 175 50 218 234 50 217 234 62 100 50 219
61764 - 234 201 60 50 217 234 71 62 22 215 62 11 215 120
61778 - 215 62 51 50 143 92 62 38 215 62 39 215 33 60
61792 - 0 17 3 0 205 181 3 201 42 213 234 43 34 213
61806 - 234 124 181 192 42 215 234 102 110 126 230 15 71 237
61820 - 106 58 120 92 173 111 16 247 126 230 63 103 126 50
61834 - 215 234 124 230 15 103 34 213 234 62 1 50 218 234
61848 - 201 58 206 234 167 202 66 242 254 1 40 56 42 207
61862 - 234 43 34 207 234 124 181 192 175 50 206 234 62 22
61876 - 215 175 215 58 205 234 215 62 36 50 143 92 62 32
```



```
61890 - 215 62 32 215 62 22 215 62 1 215 58 205 234 215
61904 - 62 36 50 143 92 62 32 215 62 32 215 201 237 95
61918 - 71 42 120 92 110 102 126 144 71 35 126 128 230 127
61932 - 6 1 254 25 56 20 6 8 254 50 56 14 6 15
61946 - 254 75 56 8 6 22 254 100 56 2 6 29 14 0
61960 - 197 205 131 37 205 213 45 193 254 36 192 120 50 205
61974 - 234 62 22 215 175 215 120 215 62 32 50 143 92 62
61988 - 41 215 62 42 215 62 22 215 62 1 215 120 215 62
62002 - 32 50 143 92 62 43 215 62 44 215 62 2 50 206
62016 - 234 201 42 209 234 43 34 209 234 124 181 192 42 211
62030 - 234 110 102 126 230 10 71 237 106 58 120 92 79 125
62044 - 169 111 16 245 124 230 63 103 34 211 234 124 230 31
62058 - 246 7 103 34 209 234 34 207 234 62 1 50 206 234
62072 - 201 237 75 221 234 205 131 37 205 213 45 71 58 221
62086 - 234 254 1 40 44 120 254 9 40 33 58 221 234 254
62100 - 11 56 7 120 230 7 254 4 32 19 58 223 234 254
62114 - 9 40 12 58 221 234 254 11 216 58 224 234 254 32
62128 - 200 62 1 50 199 234 201 58 223 234 254 36 32 243
62142 - 58 222 234 254 3 48 4 62 1 24 26 254 10 48
62156 - 4 62 8 24 18 254 17 48 4 62 15 24 10 254
62170 - 24 48 4 62 22 24 2 62 29 71 62 22 215 175
62184 - 215 120 215 62 116 50 143 92 62 45 215 62 46 215
62198 - 62 22 215 62 1 215 120 215 62 47 215 62 59 215
62212 - 1 21 16 237 67 221 234 62 63 50 225 234 205 98
62226 - 246 205 251 237 205 24 248 58 200 234 60 50 200 234
62240 - 254 5 192 62 1 50 201 234 221 33 46 236 6 13
62254 - 221 110 0 221 102 1 221 126 2 221 86 3 95 221
62268 - 35 221 35 221 35 221 35 221 229 197 213 229 120 254
62282 - 6 56 8 33 88 3 17 62 0 24 6 33 203 2
62296 - 17 73 0 205 181 3 225 209 205 181 3 193 221 225
62310 - 16 198 201 58 231 234 183 200 42 226 234 254 1 32
62324 - 3 35 24 1 43 34 226 234 175 50 231 234 42 226
62338 - 234 229 6 3 205 106 237 193 205 43 45 205 227 45
62352 - 201 58 196 234 61 50 196 234 183 192 62 100 50 196
62366 - 234 42 197 234 126 183 32 15 43 34 197 234 125 254
62380 - 228 32 6 62 1 50 199 234 201 6 4 126 203 39
62394 - 119 36 16 249 201 58 199 234 183 200 175 50 199 234
62408 - 58 228 234 61 50 228 234 205 229 236 205 47 238 62
62422 - 8 215 58 223 234 230 248 246 4 50 143 92 62 40
62436 - 215 33 100 0 6 100 197 17 2 0 229 205 181 3
62450 - 225 6 0 237 95 79 9 193 16 238 205 47 238 205
62464 - 142 237 1 21 16 237 67 221 234 62 63 50 225 234
62478 - 58 228 234 183 40 7 205 251 237 205 24 248 201 62
62492 - 1 50 190 234 6 100 118 16 253 201 221 33 110 234
62506 - 205 63 244 221 33 118 234 205 63 244 221 33 126 234
62520 - 205 63 244 221 33 134 234 221 126 0 61 221 119 0
62534 - 183 192 221 126 1 221 119 0 58 221 234 221 190 7
62548 - 32 3 205 47 238 221 110 3 221 102 4 221 94 5
62562 - 221 86 6 221 126 2 183 32 5 205 216 240 24 3
62576 - 205 177 240 58 221 234 221 190 7 192 205 251 237 201
62590 - 221 33 96 234 205 137 244 221 33 103 234 221 126 0
62604 - 61 221 119 0 183 192 221 126 1 221 119 0 58 221
62618 - 234 221 190 6 32 16 58 222 234 183 32 6 62 1
62632 - 50 199 234 201 61 50 222 234 221 110 2 221 102 3
62646 - 221 94 4 221 86 5 205 216 240 201 205 102 236 205
62660 - 251 237 221 33 142 234 205 94 239 221 33 163 234 205
62674 - 94 239 205 38 244 205 126 244 205 153 241 205 255 240
62688 - 205 69 238 205 121 242 205 191 243 205 105 243 205 145
62702 - 243 205 232 245 205 71 246 205 137 246 58 190 234 183
62716 - 192 24 199 33 0 61 17 188 252 1 0 3 237 176
62730 - 33 188 251 34 54 92 33 52 248 17 196 252 1 120
62744 - 0 237 176 17 148 253 1 48 0 237 176 17 148 254
```




```
62758 - 1 152 0 237 176 33 96 234 17 120 230 1 200 0
62772 - 237 176 62 2 205 1 22 251 62 2 205 155 34 62
62786 - 7 50 141 92 6 24 205 68 14 62 4 50 143 92
62800 - 17 77 235 1 92 0 205 60 32 62 71 50 143 92
62814 - 17 169 235 1 35 0 205 60 32 62 7 50 143 92
62828 - 17 204 235 1 41 0 205 60 32 62 6 50 143 92
62842 - 17 245 235 1 41 0 205 60 32 62 95 50 143 92
62856 - 17 30 236 1 16 0 205 60 32 62 7 50 143 92
62870 - 237 75 229 234 205 43 45 205 227 45 205 232 245 205
62884 - 71 246 30 100 1 254 253 237 120 203 79 40 5 29
62898 - 32 244 24 234 42 229 234 229 58 204 234 245 17 96
62912 - 234 33 120 230 1 200 0 237 176 241 50 204 234 225
62926 - 34 229 234 205 192 244 42 229 234 237 91 226 234 167
62940 - 237 82 242 54 245 237 83 229 234 195 54 245 58 203
62954 - 234 61 50 203 234 183 192 58 220 234 230 4 203 39
62968 - 203 39 71 62 70 144 50 203 234 58 204 234 183 32
62982 - 9 1 232 3 11 120 177 32 251 201 58 202 234 60
62996 - 50 202 234 254 120 32 5 62 2 50 202 234 230 126
63010 - 15 95 22 0 33 16 235 25 94 83 203 43 203 43
63024 - 123 130 95 83 62 2 1 3 0 211 254 29 32 3
63038 - 238 24 90 16 246 13 32 243 201 1 254 251 58 195
63052 - 234 95 237 120 230 16 187 200 50 195 234 183 200 58
63066 - 204 234 238 1 50 204 234 201 33 220 0 229 205 159
63080 - 243 225 43 229 17 2 0 205 181 3 62 1 50 231
63094 - 234 205 105 243 58 199 234 183 40 230 225 205 142 237
63108 - 175 50 199 234 201 58 201 234 183 200 58 204 234 245
63122 - 42 226 234 229 42 229 234 229 58 228 234 245 58 220
63136 - 234 33 120 230 17 96 234 1 200 0 237 176 50 220
63150 - 234 241 50 228 234 225 34 229 234 225 34 226 234 241
63164 - 50 204 234 58 220 234 60 50 220 234 254 6 32 7
63178 - 175 50 220 234 195 106 247 79 87 203 34 203 34 203
63192 - 34 203 34 221 33 110 234 6 4 221 126 1 146 221
63206 - 119 1 221 119 0 122 17 8 0 221 25 87 16 237
63220 - 221 33 96 234 6 2 221 126 1 146 221 119 1 221
63234 - 119 0 122 17 7 0 221 25 87 16 237 221 33 142
63248 - 234 6 2 221 126 2 146 221 119 2 221 119 0 221
63262 - 33 163 234 221 126 2 146 221 119 2 221 119 0 121
63276 - 230 4 15 87 58 184 234 130 50 184 234 203 42 58
63290 - 185 234 146 50 185 234 6 0 58 220 234 254 3 56
63304 - 33 32 6 6 3 62 2 24 12 31 203 16 203 39
63318 - 135 203 32 128 71 121 61 50 187 234 50 189 234 120
63332 - 50 186 234 50 188 234 225 195 192 244 58 120 92 111
63346 - 237 95 103 126 172 103 230 63 102 133 172 111 110 126
63360 - 230 7 201 0 0 0 0 0 33 186 234 34 133 247
63374 - 175 50 132 247 62 13 50 131 247 62 1 50 135 247
63388 - 6 4 197 24 61 205 110 247 42 133 247 78 254 2
63402 - 48 3 13 24 5 254 4 40 1 12 58 132 247 129
63416 - 50 132 247 254 30 56 31 58 131 247 60 60 50 131
63430 - 247 58 135 247 238 1 50 135 247 175 50 132 247 42
63444 - 133 247 35 34 133 247 193 16 193 201 62 22 215 58
63458 - 131 247 215 58 132 247 215 58 132 247 198 2 50 132
63472 - 247 205 110 247 230 3 60 60 60 50 143 92 58 135
63486 - 247 183 32 4 22 93 24 2 22 101 205 110 247 230
63500 - 3 203 39 130 245 215 241 60 215 195 161 247 62 22
63514 - 215 62 13 215 175 50 143 92 215 6 224 197 62 32
63528 - 215 193 16 249 205 136 247 201 139 247 201 201 31 63
63542 - 97 127 127 97 63 31 255 255 225 255 15 255 248 255
63556 - 248 228 194 66 194 194 228 248 223 127 127 255 255 127
63570 - 127 223 96 192 216 252 252 216 192 96 0 0 0 0
63584 - 0 3 100 152 24 44 32 16 8 8 144 96 129 66
63598 - 60 36 36 60 66 129 0 0 0 0 0 0 1 7
63612 - 0 0 0 7 31 126 250 232 15 222 250 57 221 31
63626 - 63 192 160 128 20 95 255 254 224 0 0 0 56 124
```



```
63640 - 158 159 127 63 0 0 28 62 121 249 254 252 127 127
63654 - 111 51 28 15 3 0 254 254 246 204 56 240 192 0
63668 - 24 60 126 255 60 60 60 60 8 12 254 255 255 254
63682 - 12 8 60 60 60 60 255 126 60 24 153 165 102 60
63696 - 60 126 189 66 67 164 126 121 121 126 164 67 66 189
63710 - 126 60 60 102 165 153 194 37 126 158 158 126 37 194
63724 - 15 17 33 255 255 255 60 24 255 255 255 255 255 255
63738 - 60 24 0 0 0 31 127 255 56 16 0 112 136 6
63752 - 255 255 28 8 15 17 33 255 255 255 60 24 0 170
63766 - 85 255 255 255 120 48 0 1 3 126 191 255 60 24
63780 - 0 224 80 172 255 255 60 24 255 255 255 255 255 255
63794 - 60 24 240 136 132 255 255 255 60 24 0 14 17 96
63808 - 255 255 56 16 0 0 0 248 254 255 28 8 0 85
63822 - 170 255 255 255 30 12 240 136 132 255 255 255 60 24
63836 - 0 7 10 53 255 255 60 24 0 128 192 126 253 255
63850 - 60 24 16 48 127 255 255 127 48 16 0 0 0 0
63864 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63878 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63892 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63906 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63920 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63934 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63948 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63962 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63976 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
63990 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
64004 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
64018 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
64032 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
64046 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
64060 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0
64074 - 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10 REM PROGRAMM 2
20 CLEAR 50000: LOAD ""CODE : RANDOMIZE USR 62719
```

LASER 110 – 4 KByte RAM. Dieser „persönliche Computer" eignet sich als Helfer beim Studium, am Arbeitsplatz oder bei Analysen und Statistiken. Mit seinen Peripheriegeräten erfüllt er alle Ansprüche an einen Home-Computer.

Tödliche Strahlen greifen die Stadt an.

Im Spiel Donovan müssen Sie versuchen, mit einer Laserkanone tödliche Strahlen abzuwehren. Dazu wird die Zieloptik der Kanone auf die Spitze der Strahlen gelenkt. Ist die Optik maximal 10 Bildpunkte von der Strahlenspitze entfernt, (abhängig vom Schwierigkeitsgrad), kann abgedrückt werden. Der Angreifer wird zerstört und der Spieler erhält 10 Punkte. Pro abgeschossenen Strahl erhält man 10 Runden-Punkte. Pro Runde erscheinen 10 Strahlen. Nach jeder 6. überstandenen Runde erhält man einen Bonus. – Die Strahlen zielen immer auf eine der Städte oder auf ihre Laserstation. Hat ein Strahl sein Ziel getroffen, ist es vernichtet. Wird die Station getroffen, werden Punkte und Energieeinheiten abgezogen. Der Energievorrat ist begrenzt. Ist die gesamte Energie verbraucht, kann nicht mehr geschossen werden.
Die Steuerung des Spiels erfolgt über die Tastatur mit folgenden Tasten: ›I‹ hoch, ›M‹ runter, ›J‹ links, ›K‹ rechts. Gefeuert wird mit der Taste ›A‹.
Gewählt werden kann zwischen 10 verschiedenen Spielstärken. Außerdem besteht die Möglichkeit, den Ton während des Spiels abzustellen.

Programmablauf:
Nach dem Laden erscheint das Menue. Dort kann man wählen zwischen:
- Spiel beenden
- Anleitung lesen
- Ton an- oder abstellen
- Schwierigkeitsgrad verändern
- Spiel beginnen

Hat man sich für „Spiel beginnen" entschieden, wird auf hochauflösende Grafik umgeschaltet und das Spielfeld aufgebaut. Danach beginnt das eigentliche Spiel.

Nach jeder überstandenen Welle wird rechts unten angezeigt, die wievielte Welle jetzt kommt („WAVE").
Nach jeder sechsten Welle erhält man einen Bonus, angezeigt durch das Wort „BONUS" in der rechten unteren Ecke.

Wenn alle sechs Städte vernichtet sind, erscheint die Meldung „GAME OVER", und nach einer kurzen Pause befindet man sich wieder im Menue. Zwischendurch ertönen immer wieder kleine Melodien.

```
1   REM "*************************************************************"
3   REM "*                    D  O  N  O  V  A  N                    *"
4   REM "*                                                           *"
5   REM "*                   Copyright 1984 by                       *"
6   REM "*                                                           *"
7   REM "*             N O R B E R T   F A E H R M A N N             *"
8   REM "*                                                           *"
9   REM "*************************************************************"
10  :
20   GOTO 3100
30   REM    *** HAUPTPROGRAMM ***
40   GOSUB 1090:WA% = WA% + 1: IF WA% > 99 THEN WA% = 99
50  X = 215:Y = 181:HI% = P:P = WA%: POKE FA%,127: GOSUB 1830: GOSUB 1730
60   POKE 768,40: POKE 769,100: CALL 770: POKE 768,50: POKE 769,255: CALL 7
     70: FOR I = 1 TO 700: NEXT
70  X = 215:Y = 181:P = HI%: GOSUB 1090
80   IF WA% / 6 =  INT (WA% / 6) THEN  GOSUB 1100
90   GOSUB 1730
100  FOR AN = 1 TO 10
110 A% =  RND (1) * 6 + 1.5: IF ST%(A%,1) = 0 THEN 110
120 CO% =  RND (1) * 6 + 1.5: IF CO% = 5 OR CO% = 2 OR CO% = 4 THEN 120
130 SY% =  RND (1) * 234 + 10
140 B1 = SY%
150 ST = ST%(A%,2) - SY%
160 ST = ST / 146 * LE%
170 LX% = 138:LY% = 140
180  POKE FA%,127: XDRAW 1 AT LX%,LY%
190  FOR Y = 15 TO 160 STEP LE%
200  POKE 768,100: POKE 769,2: CALL 770
210  FOR I = 1 TO 3
220 KB% =  PEEK (TA%)
230  IF KB% < FI% OR KB% > UN% THEN 310
240  POKE FA%,127: XDRAW 1 AT LX%,LY%
250  IF KB% = RE% THEN  IF LX% < 246 THEN LX% = LX% + VI%: GOTO 300
260  IF KB% = LI% THEN  IF LX% > 11 THEN LX% = LX% - VI%: GOTO 300
270  IF KB% = OB% THEN  IF LY% > 49 THEN LY% = LY% - VI%: GOTO 300
280  IF KB% = UN% THEN  IF LY% < 141 THEN LY% = LY% + VI%: GOTO 300
290  IF KB% = FI% THEN 460
300  XDRAW 1 AT LX%,LY%
310  NEXT I
320  HCOLOR= CO%
```

```
330  HPLOT SY%,15 TO B1,Y
340 B1 = B1 + ST
350  NEXT Y
360  GOSUB 740
370  FOR K = 1 TO 6: IF ST%(K,1) THEN 400
380  NEXT K
390  GOTO 2730
400  NEXT AN
410  FOR K = 1 TO 6: IF ST%(K,1) THEN X = ST%(K,2) - 7:Y = 161: GOSUB 2040
     : POKE 768,(220 - (K * 10)): POKE 769,100: CALL 770:P = P + 100 * LE%
     : GOSUB 1730
420  NEXT
430 LE% = LE% + 1: IF LE% > 10 THEN LE% = 10
440  FOR I = 1 TO 1000: NEXT : GOTO 40
450  REM    *** SCHUSS ***
460  POKE FA%,127
470  POKE  - 16368,0
480  IF FU% = 0 THEN 300
490  HPLOT 138,159 TO LX%,LY%
500  DRAW 5 AT LX%,LY%: FOR J = 60 TO 30 STEP  - 2: POKE 768,J: POKE 769,4
     : CALL 770: NEXT : POKE FA%,0: DRAW 5 AT LX%,LY%: DRAW 1 AT LX%,LY%
510  HPLOT LX%,LY% TO 138,159
520  HPLOT 138,159 TO LX%,LY%
530  IF  ABS (LY% - Y) < 11 - LE% * .3 THEN 560
540 FU% = FU% - 1: HPLOT FU% + 16,184 TO FU% + 16,186
550  GOTO 300
560  IF  ABS (B1 - LX%) < 11 - LE% * .3 THEN 580
570  GOTO 540
580 X = B1: GOSUB 600
590  GOTO 370
600  REM    *** TREFFER ! ***
610  POKE FA%,127
620  DRAW 3 AT X,Y: GOSUB 1010: XDRAW 3 AT X,Y: DRAW 4 AT X,Y - 2: GOSUB 1
     010: XDRAW 4 AT X,Y - 2: DRAW 5 AT X + 3,Y - 2: GOSUB 1010: XDRAW 5 AT
     X + 3,Y - 2: DRAW 6 AT X - 3,Y: GOSUB 1010: POKE FA%,0: DRAW 6 AT X -
     3,Y
630  FOR I =  - 10 TO 10
640  HPLOT SY% + I,22 TO B1 + I,Y - LE%
650  HPLOT SY% + I,22 TO B1 + I,Y
660  HPLOT SY% + I,22 TO B1 + I,Y + LE%
670  NEXT
680  FOR I =  - 10 TO 10
690  HPLOT X - I,Y + 10 TO X + I,Y - 10
700  NEXT
710 P = P + 10 * LE%: GOSUB 1720
720  RETURN
730  REM    *** STRAHL ERREICHT BODEN ***
740 X = ST%(A%,2):Y = 164
750  POKE FA%,0: DRAW 1 AT LX%,LY%
760  POKE FA%,127: DRAW 3 AT X,Y + 2: GOSUB 1030: XDRAW 3 AT X,Y + 2: DRAW
     6 AT X + 2,Y + 2: GOSUB 1030: XDRAW 6 AT X + 2,Y + 2
770  DRAW 5 AT X,Y: GOSUB 1030: XDRAW 5 AT X,Y: DRAW 7 AT X - 2,Y + 2: GOSUB
     1030: POKE FA%,0: DRAW 7 AT X - 2,Y + 2
780  FOR I =  - 10 TO 14
790  POKE 768,245: POKE 769,4: CALL 770
800  HPLOT X - 10,Y - I TO X + 10,Y + I - 4
810  NEXT
820  FOR I =  - 7 TO 9
830  POKE 768,245: POKE 769,4: CALL 770
840  HPLOT SY% + I,20 TO B1 + I,161
850  HPLOT B1,161 TO B1 + I,174
860  NEXT
870  POKE 768,255: POKE 769,5: CALL 770
880  FOR I = 160 TO 174: HPLOT X - 10,I TO X + 14,I
```



```
890  NEXT
900  IF A% < 7 THEN 970
910  GOSUB 980:P = P - 50 * LE%: GOSUB 1090: IF P < 0 THEN P = 0
920  GOSUB 1730: FOR I = 1 TO 200: NEXT : POKE FA%,0
930  FOR I = 1 TO 50 - LE% * 4:FU% = FU% - 1: IF FU% < 0 THEN FU% = 0: GOTO
     960
940  HPLOT FU% + 16,184 TO FU% + 16,186
950  POKE 768,25 + I: POKE 769,5: CALL 770: NEXT
960  FOR I = 1 TO 400: NEXT : GOSUB 980: FOR I = 1 TO 600: NEXT : GOSUB 10
     20: GOSUB 2250: RETURN
970 ST%(A%,1) = 0
980  POKE 768,200: POKE 769,100: CALL 770: POKE 768,220: POKE 769,100: CALL
     770: POKE 768,245: POKE 769,200: CALL 770
990  RETURN
1000  REM  *** SOUNDROUTINEN ***
1010  FOR J = 30 TO 50: POKE 768,J: POKE 769,4: CALL 770: NEXT : RETURN
1020  FOR J = 250 TO 90 STEP  - 3: POKE 768,J: POKE 769,5: CALL 770: NEXT
     : RETURN
1030  FOR J = 160 TO 200 STEP 5: POKE 768,J: POKE 769,8: CALL 770: NEXT : RETURN

1040  FOR J = 1 TO 23: POKE 768,(255 - T%(J)): POKE 769,L%(J): CALL 770: NEXT
     : RETURN
1050  FOR J = 1 TO 19: POKE 768,TS%(J): POKE 769,LS%(J): CALL 770: NEXT : RETURN
1060  FOR J = 1 TO 15: POKE 768,LT%(J): POKE 769,TT%(J): CALL 770: NEXT : RETURN

1070  FOR I = 120 TO 60 STEP  - 3: POKE 768,I: POKE 769,5: CALL 770: NEXT
     : RETURN
1080  REM *** RECHTEN UNTEREN TEIL LOESCHEN ***
1090  POKE FA%,0: FOR I = 180 TO 190: HPLOT 270,I TO 210,I: NEXT : RETURN

1100  REM *** BONUS ***
1110  GOSUB 1090
1120 X = 215:Y = 181: FOR I = 1 TO 5: POKE FA%,0: GOSUB 1940
1130  POKE FA%,127: GOSUB 1940: POKE 768,70: POKE 769,60 - I * 10: CALL 77
     0: NEXT
1140  FOR I = 1 TO 900: NEXT : FOR I = 1 TO 3: POKE 768,80: POKE 769,100: CALL
     770: NEXT : POKE 768,100: POKE 769,255: CALL 770: FOR I = 1 TO 500: NEXT

1150  GOSUB 1090
1160  FOR K = 1 TO 6: IF ST%(K,1) = 0 THEN ST%(K,1) = 1:X = ST%(K,2) - 7:Y
      = 161:LE% = LE% - 1: GOSUB 2040: GOSUB 1070:LE% = LE% + 1: GOTO 1180

1170  NEXT :P = P + LE% * 60
1180  RETURN
1190  REM  *** ZAHLEN FUER SCORE ***
1200  FOR I = 181 TO 189 STEP 8: HPLOT X + 2,I TO X + 4,I
1210  NEXT
1220  FOR J = 1 TO 5 STEP 4
1230  FOR I = 182 TO 186 STEP 4: HPLOT X + J,I TO X + J,I + 2
1240  NEXT
1250  NEXT
1260  RETURN
1270  HPLOT X + 5,182 TO X + 5,184
1280  HPLOT X + 5,186 TO X + 5,188
1290  RETURN
1300  FOR I = 181 TO 189 STEP 4: HPLOT X + 2,I TO X + 4,I
1310  NEXT
1320  HPLOT X + 5,182 TO X + 5,184
1330  HPLOT X + 1,186 TO X + 1,188
1340  RETURN
1350  FOR I = 181 TO 189 STEP 4: HPLOT X + 2,I TO X + 4,I
1360  NEXT
1370  HPLOT X + 5,182 TO X + 5,184
1380  HPLOT X + 5,186 TO X + 5,188
```



```
1390  RETURN
1400  HPLOT X + 1,182 TO X + 1,184
1410  HPLOT X + 5,182 TO X + 5,184
1420  HPLOT X + 2,185 TO X + 4,185
1430  HPLOT X + 5,186 TO X + 5,188
1440  RETURN
1450  FOR I = 181 TO 189 STEP 4: HPLOT X + 2,I TO X + 4,I
1460  NEXT
1470  HPLOT X + 1,182 TO X + 1,184
1480  HPLOT X + 5,186 TO X + 5,188
1490  RETURN
1500  FOR I = 181 TO 189 STEP 4: HPLOT X + 2,I TO X + 4,I
1510  NEXT
1520  HPLOT X + 1,182 TO X + 1,184
1530  HPLOT X + 5,186 TO X + 5,188
1540  HPLOT X + 1,186 TO X + 1,188
1550  RETURN
1560  HPLOT X + 2,181 TO X + 4,181
1570  HPLOT X + 5,182 TO X + 5,184
1580  HPLOT X + 5,186 TO X + 5,188
1590  RETURN
1600  FOR I = 181 TO 189 STEP 4: HPLOT X + 2,I TO X + 4,I
1610  NEXT
1620  FOR J = 1 TO 5 STEP 4
1630  FOR I = 182 TO 186 STEP 4: HPLOT X + J,I TO X + J,I + 2
1640  NEXT
1650  NEXT
1660  RETURN
1670  FOR I = 181 TO 189 STEP 4: HPLOT X + 2,I TO X + 4,I
1680  NEXT
1690  HPLOT X + 5,182 TO X + 5,184
1700  HPLOT X + 5,186 TO X + 5,188
1710  HPLOT X + 1,182 TO X + 1,184: RETURN
1720  REM *** PUNKTE PLOTTEN ***
1730 P$ =  STR$ (P)
1740 X = 263 - 7 *  LEN (P$)
1750 L =  VAL ( MID$ (P$,1,1)) + 1
1760  POKE FA%,0
1770  HPLOT X + 1,181 TO X + 5,181 TO X + 5,189 TO X + 1,189 TO X + 1,181
1780  HPLOT X + 2,185 TO X + 4,185
1790  POKE FA%,127: ON L GOSUB 1200,1270,1300,1350,1400,1450,1500,1560,160
   0,1670
1800  IF  LEN (P$) < 2 THEN  RETURN
1810 P$ =  RIGHT$ (P$, LEN (P$) - 1): GOTO 1740
1820  REM  *** 'WAVE' PLOTTEN ***
1830  HPLOT X,Y TO X,Y + 6
1840  HPLOT X + 1,Y + 7 TO X + 5,Y + 7
1850  HPLOT X + 3,Y + 7 TO X + 3,Y + 3
1860  HPLOT X + 6,Y + 6 TO X + 6,Y
1870  HPLOT X + 9,Y + 7 TO X + 9,Y TO X + 14,Y TO X + 14,Y + 7
1880  HPLOT X + 9,Y + 3 TO X + 14,Y + 3
1890  HPLOT X + 17,Y TO X + 17,Y + 6: HPLOT X + 21,Y TO X + 21,Y + 6: HPLOT
   X + 18,Y + 7 TO X + 20,Y + 7
1900  HPLOT X + 28,Y TO X + 24,Y TO X + 24,Y + 7 TO X + 28,Y + 7
1910  HPLOT X + 24,Y + 3 TO X + 27,Y + 3
1920  RETURN
1930  REM *** 'BONUS' PLOTTEN ***
1940  HPLOT X,Y TO X,Y + 7 TO X + 3,Y + 7
1950  HPLOT X + 4,Y + 6 TO X + 4,Y + 4: HPLOT X + 4,Y + 2 TO X + 4,Y + 1
1960  HPLOT X + 1,Y + 3 TO X + 3,Y + 3: HPLOT X + 1,Y TO X + 3,Y
1970  HPLOT X + 7,Y TO X + 7,Y + 7 TO X + 11,Y + 7 TO X + 11,Y TO X + 7,Y
1980  HPLOT X + 14,Y + 7 TO X + 14,Y TO X + 18,Y TO X + 18,Y + 7
1990  HPLOT X + 21,Y TO X + 21,Y + 7 TO X + 25,Y + 7 TO X + 25,Y
2000  HPLOT X + 32,Y TO X + 28,Y TO X + 28,Y + 4 TO X + 32,Y + 4 TO X + 32
```



```
,Y + 7 TO X + 28,Y + 7
2010  HPLOT X + 38,Y TO X + 38,Y + 5: HPLOT X + 38,Y + 7
2020  RETURN
2030  REM *** STADT ZEICHNEN ***
2040 CO% =  ABS (LE% - 5) + 1: IF CO% = 4 THEN CO% = 5
2050  HCOLOR= CO%
2060  HPLOT X + 7,Y TO X + 7,Y + 1
2070  FOR I = 2 TO 3: HPLOT X + 6,Y + I TO X + 8,Y + I
2080  NEXT
2090  FOR I = 4 TO 8: HPLOT X + 5,Y + I TO X + 9,Y + I
2100  NEXT
2110  FOR I = 9 TO 10: HPLOT X,Y + I TO X + 9,Y + I
2120  NEXT
2130  FOR I = 11 TO 13: HPLOT X,Y + I TO X + 14,Y + I
2140  NEXT
2150  FOR I = 12 TO 13: HPLOT X + I,Y + 5 TO X + I,Y + 10
2160  NEXT
2170  HPLOT X + 2,Y + 7
2180  HPLOT X + 1,Y + 8 TO X + 3,Y + 8
2190  POKE FA%,0: HPLOT X + 2,Y + 10
2200  HPLOT X + 7,Y + 4 TO X + 7,Y + 5
2210  FOR I = 11 TO 13 STEP 2: HPLOT X + I,Y + 12
2220  NEXT
2230  RETURN
2240  REM      *** LASERBASIS ZEICHNEN ***
2250  POKE FA%,127
2260  HPLOT 138,160 TO 138,162
2270  HPLOT 139,160 TO 139,162
2280  FOR I = 163 TO 165
2290  HPLOT 137,I TO 140,I
2300  NEXT
2310  FOR I = 166 TO 170
2320  HPLOT 136,I TO 141,I
2330  NEXT
2340  POKE FA%,0: HPLOT 138,166 TO 139,166
2350  HPLOT 137,167 TO 137,168
2360  HPLOT 140,167 TO 140,168
2370  HPLOT 138,169 TO 139,169
2380  POKE FA%,127
2390  FOR I = 131 TO 135: HPLOT I,175 TO I + 4,171
2400  NEXT
2410  FOR I = 146 TO 142 STEP  - 1: HPLOT I,175 TO I - 4,171
2420  NEXT
2430  RETURN
2440  REM *** WOLKEN ZEICHNEN ***
2450  FOR I = 1 TO 5: DRAW ( INT ( RND (1) * 5) + 2) AT X + I,Y
2460  POKE FA%, INT ( RND (1) * 255): NEXT
2470  POKE FA%,0: DRAW 6 AT X - 2,Y
2480  RETURN
2490  REM *** ENERGIEBAND ZEICHNEN ***
2500  POKE FA%,127
2510  FOR I = 181 TO 189 STEP 4: HPLOT 10,I TO 12,I
2520  NEXT
2530  HPLOT 9,181 TO 9,189
2540  HPLOT 15,187 TO 16 + FU%,187 TO 16 + FU%,183 TO 15,183 TO 15,187
2550  POKE FA%,85
2560  FOR I = 184 TO 186: HPLOT 16,I TO 15 + FU%,I
2570  NEXT
2580  RETURN
2590  REM      *** 'DONOVAN' IN GROSSCHRIFT PLOTTEN... ***
2600 X = 30:Y = 56
2610  HCOLOR= FA: FOR J = 1 TO 7
2620  FOR I = 0 TO 2
2630  ON SK(J) GOSUB 2840,2860,2890,2910,2940,2960,2980,3000,3020
```



```
2640  NEXT
2650 X = X + 30
2660  NEXT
2670  RETURN
2680  REM   *** ...UND WIEDER LOESCHEN ***
2690  POKE FA%,0: FOR I = 1 TO 60: HPLOT 30 + I,54 TO 30 + I,78: HPLOT 235
      - I,54 TO 235 - I,78: NEXT
2700  FOR I = 1 TO 16: HPLOT 30,54 + I TO 240,54 + I: HPLOT 30,80 - I TO 2
     40,80 - I: NEXT
2710  RETURN
2720  REM *** 'GAME OVER' IN GROSSCHRIFT PLOTTEN... *
2730 X = 60:Y = 48: POKE FA%,127: FOR J = 1 TO 8
2740  FOR I = 0 TO 2
2750  ON SL(J) GOSUB 2840,2860,2890,2910,2940,2960,2980
2760  NEXT I
2770 X = X + 40: IF J = 4 THEN X = 60:Y = 88
2780  NEXT
2790  GOSUB 1040
2800  REM *** ...UND NEUES SPIEL ***
2810  FOR I = 1 TO 6:ST%(I,1) = 1: NEXT :LE% = 0:WA% = 0: IF P >  = HS THEN
     HS = P:HL = R1
2820  FOR I = 1 TO 2000: NEXT : HGR2 : TEXT : GOTO 3180
2830  REM *** BUCHSTABEN FUER SCHRIFTZUEGE ***
2840  HPLOT X + 19 + I,Y + 1 - I TO X + 2 + I,Y + 1 - I TO X + 2 + I,Y + 2
     2 - I TO X + 19 + I,Y + 22 - I TO X + 19 + I,Y + 12 - I TO X + 7 + I,
     Y + 12 - I
2850  RETURN
2860  HPLOT X + 2 + I,Y + 22 - I TO X + 10 + I,Y + 1 - I TO X + 11 + I,Y +
     1 - I TO X + 19 + I,Y + 22 - I
2870  HPLOT X + 6 + I,Y + 12 - I TO X + 15 + I,Y + 12 - I
2880  RETURN
2890  HPLOT X + 2 + I,Y + 22 - I TO X + 2 + I,Y + 1 - I TO X + 10 + I,Y +
     12 - I TO X + 11 + I,Y + 12 - I TO X + 19 + I,Y + 1 - I TO X + 19 + I
     ,Y + 22 - I
2900  RETURN
2910  HPLOT X + 19 + I,Y + 1 - I TO X + 2 + I,Y + 1 - I TO X + 2 + I,Y + 2
     2 - I TO X + 19 + I,Y + 22 - I
2920  HPLOT X + 2 + I,Y + 12 - I TO X + 15 + I,Y + 12 - I
2930  RETURN
2940  HPLOT X + 2 + I,Y + 1 - I TO X + 2 + I,Y + 22 - I TO X + 19 + I,Y +
     22 - I TO X + 19 + I,Y + 1 - I TO X + 2 + I,Y + 1 - I
2950  RETURN
2960  HPLOT X + 2 + I,Y + 1 - I TO X + 10 + I,Y + 22 - I TO X + 11 + I,Y +
     22 - I TO X + 19 + I,Y + 1 - I
2970  RETURN
2980  HPLOT X + 2 + I,Y + 22 - I TO X + 2 + I,Y + 1 - I TO X + 15 + I,Y +
     1 - I TO X + 19 + I,Y + 5 - I TO X + 19 + I,Y + 8 - I TO X + 15 + I,Y
      + 12 - I TO X + 6 + I,Y + 12 - I TO X + 19 + I,Y + 22 - I
2990  RETURN
3000  HPLOT X + 2 + I,Y + 22 - I TO X + 2 + I,Y + 1 - I TO X + 19 + I,Y +
     22 - I TO X + 19 + I,Y + 1 - I
3010  RETURN
3020  HPLOT X + 2 + I,Y + 1 - I TO X + 15 + I,Y + 1 - I TO X + 19 + I,Y +
     5 - I TO X + 19 + I,Y + 18 - I TO X + 15 + I,Y + 22 - I TO X + 2 + I,
     Y + 22 - I TO X + 2 + I,Y + 1 - I
3030  RETURN
3040  REM  *** HIGHSCORE DRUCKEN ***
3050  VTAB 9: HTAB 30: PRINT HS
3060  HTAB 20: VTAB 13: PRINT LE%
3070  VTAB 13: HTAB 30: PRINT HL
3080  RETURN
3090  REM   *** EINLEITUNG ***
3100  TEXT : HOME : VTAB 3: INVERSE : PRINT "            D O N O V A N
          "
```



```
3110  NORMAL : VTAB 22: PRINT " Copyright 1984 by   ";: INVERSE : PRINT "N
     ORBERT FAEHRMANN": NORMAL
3120  VTAB 7: INVERSE : HTAB 5: PRINT " ESC ";: NORMAL : PRINT " = no Game
     ": PRINT : INVERSE : HTAB 5: PRINT "  S  ";: NORMAL : PRINT " = Sound
     ": GOSUB 3260
3130  PRINT : HTAB 5: INVERSE : PRINT " RET ";: NORMAL : PRINT " = start G
     ame": PRINT : HTAB 5: INVERSE : PRINT "0...9";: NORMAL : PRINT " = Sp
     eed (";LE%;")"
3140  PRINT : INVERSE : HTAB 5: PRINT "  I  ";: NORMAL : PRINT " = Instruc
     tions"
3150  VTAB 7: HTAB 30: INVERSE : PRINT "HI-SCORE": VTAB 11: HTAB 30: PRINT
     "LEVEL": NORMAL
3160  IF PO% = 0 THEN PO% = 1: GOSUB 3050: GOSUB 3470
3170  VTAB 18: HTAB 5: INVERSE : PRINT "ENTER YOUR SELECTION": NORMAL
3180  GOSUB 3050
3190  IF  PEEK (TA%) < 128 THEN 3190
3200  GET A$: IF A$ >  = "0" AND A$ <  = "9" THEN LE% =  VAL (A$): VTAB 13
     : HTAB 19: PRINT "(";LE%;")"
3210  IF A$ = "S" THEN  GOSUB 3260
3220  IF A$ =  CHR$ (27) THEN  VTAB 7: HTAB 13: INVERSE : PRINT "NO GAME":
      NORMAL : FOR I = 1 TO 1000: NEXT : HOME : END
3230  IF A$ = "I" THEN  VTAB 15: HTAB 13: INVERSE : PRINT "INSTRUCTIONS": NORMAL
 : FOR I = 1 TO 1000: NEXT : GOTO 3610
3240  IF A$ =  CHR$ (13) THEN  VTAB 11: HTAB 13: FLASH : PRINT "START GAME
     ": NORMAL : GOTO 3280
3250  GOTO 3190
3260  VTAB 9: HTAB 19: IF SO% THEN SO% = 0: POKE 770,96: PRINT "on / ";: INVERSE
 : PRINT "OFF": NORMAL : RETURN
3270  IF SO% = 0 THEN SO% = 1: POKE 770,173: INVERSE : PRINT "ON";: NORMAL
     : PRINT " / off": RETURN
3280  IF SO% THEN  GOSUB 1060: GOTO 3300
3290  FOR I = 1 TO 1500: NEXT
3300 R1 = LE%: VTAB 11: HTAB 13: PRINT "Start Game"
3310  REM   *** MACHE TITELBILD ***
3320  HGR2
3330  GOSUB 2250
3340  POKE FA%,85
3350  FOR I = 175 TO 179: HPLOT 0,I TO 279,I
3360  NEXT
3370 P = 0:FU% = 148 - (LE% * 7)
3380  GOSUB 2500
3390  FOR X = 7 TO 275 STEP 8:Y = 12 +  INT ( RND (1) * 6): GOSUB 2450
3400  NEXT
3410 Y = 161: FOR J = 1 TO 6:X = ST%(J,2) - 7: GOSUB 2040: IF PO% = 2 THEN
      GOSUB 1070
3420  NEXT : IF PO% = 2 THEN 3450
3430 FA = 2: GOSUB 2600:FA = 3: GOSUB 2600: GOSUB 1050: GOSUB 2690
3440 PO% = 2
3450 LE% = LE% + 1: GOTO 40: REM  *** RUECKSPRUNG INS HAUPTPROGRAMM ***
3460  REM  *** EINLESEN DER DATEN ***
3470  DIM T%(23),L%(23),TS%(19),LS%(19),LT%(15),TT%(15),ST%(7,2)
3480  FOR I = 28672 TO 28672 + 285: READ A%: POKE I,A%: NEXT
3490  POKE 232,0: POKE 233,112: ROT= 0: SCALE= 1
3500  FOR I = 770 TO 790: READ A%: POKE I,A%: NEXT
3510  FOR I = 1 TO 23: READ T%(I),L%(I): NEXT
3520  FOR I = 1 TO 19: READ TS%(I),LS%(I): NEXT
3530  FOR I = 1 TO 15: READ LT%(I),TT%(I): NEXT
3540  FOR I = 1 TO 7: READ SK(I): NEXT
3550  FOR I = 1 TO 8: READ SL(I): NEXT
3560  FOR I = 1 TO 7:ST%(I,1) = 1: NEXT
3570 ST%(1,2) = 26:ST%(2,2) = 61:ST%(3,2) = 96:ST%(4,2) = 182:ST%(5,2) = 2
     17:ST%(6,2) = 252:ST%(7,2) = 138
3580 TA% =  - 16384:RE% = 203:LI% = 202:OB% = 201:UN% = 205:FI% = 193:VI% =
     4:FA% = 228
```



```
3590  RETURN
3600  REM               *** SPIELANLEITUNG ***
3610  HOME
3620  PRINT "                D O N O V A N"
3630  PRINT "                ============="
3640  PRINT
3650  PRINT "Alarmstufe ROT !!! Toedliche Strahlen   greifen aus dem Welta
    ll an . Sie sind   der letzte Ueberlebende ,der die Mensch-heit vor d
    er totalen Vernichtung retten kann."
3660  PRINT : PRINT "Zerstoeren Sie die totbringenden Strah- len , bevor s
    ie die Erde erreicht haben.Sie koennen die Strahlen vernichten ,   in
    dem Sie Ihre Zieloptik so nah wie"
3670  PRINT "moeglich an die Spitze des Strahls len- ken  . Ist die Optik
    beim Schuss maximalzehn Bildpunkte von der Spitze des      Strahls en
    tfernt , so wird er zerstoert und Sie erhalten dafuer Punkte."
3680  VTAB 23: PRINT "Bitte eine Taste druecken !": WAIT  - 16384,128: POKE
     - 16368,0: HOME
3690  HOME : PRINT "               D O N O V A N"
3700  PRINT "                ============="
3710  PRINT
3720  PRINT "Die Strahlen zielen immer entweder auf  eine der Staedte oder
     die Laserstation. Trifft der Strahl auf eine Stadt , wird sie zersto
    ert . Trifft er jedoch   auf  die Laserstation , werden Ihnen Energie
     und Punkte abgezogen."
3730  PRINT "Achten Sie besonders auf das Energie-   band , denn bei jedem
     Fehlschuss wird   wird Ihnen eine Energieeinheit abge-    zogen. Ist
     die gesamte Energie ver-     braucht , koennen Sie nur"
3740  PRINT "noch tatenlos zusehen ,wie Ihre Staedte vernichtet werden ."
3750  PRINT "Das Spiel ist beendet , wenn alle sechs Staedte vernichtet si
    nd . Nach jeder     sechsten Welle erhalten Sie einen Bonus,entweder e
    ine zerstoerte Stadt oder     Punkte."
3760  VTAB 23: PRINT "Bitte eine Taste druecken !": WAIT  - 16384,128: POKE
     - 16368,0: HOME
3770  PRINT "                D O N O V A N"
3780  PRINT "                ============="
3790  PRINT
3800  PRINT "Sie steuern Ihre Zieloptik mit : "
3810  PRINT
3820  PRINT "                              I"
3830  PRINT "Sie feuern mit 'A' .     J   K"
3840  PRINT "                              M"
3850  PRINT : PRINT : PRINT "Es stehen Ihnen zehn Schwierigkeitsgradezur V
    erfuegung , von 0 (leicht) bis     9 (schwer) . ": PRINT
3860  PRINT : PRINT : PRINT : PRINT "Ich wuensche  V I E L  S P A S S  !!"

3870  VTAB 23: PRINT "Bitte eine Taste druecken !": WAIT  - 16384,128: POKE
     - 16368,0: HOME
3880  IF SO% THEN SO% = 0: GOTO 3100
3890 SO% = 1: GOTO 3100
3900  REM  *** DATAS FUER... ***
3910  REM ***... SHAPES ***
3920  DATA   7,0,16,0,30,0,44,0,89 ,0,126,0,155,0,225,0
3930  DATA  41,61,3,32,24,54,63,19,45,54,10,36,37,0
3940  DATA  28,28,36,12,12,12,21,21,              21,54,30,30,30,6,0
3950  DATA  56,36,63,42,54,21,53,46,32,37,39,33,37,63,7,40,37,59,27,56,23,
    23,36,21,18,18,51,53,13,18,45,9,41,36,33,33,39,4,24,32,39,28,59,7,0
3960  DATA    32,37,39,33,37,63,7,40,37,59,27,56,23,23,36,21,18,18,51,53,1
    3,18,45,9,41,36,33,33,39,4,24,32,39,28,59,7,0
3970  DATA    37,59,27,56,23,23,36,21,18,18,51,53,13,18,45,9,41,36,33,33,3
    9,4,24,32,39,28,59,7,0
3980  DATA    24,8,23,63,62,54,53,53,53,45,37,45,60,44,44,60,39,39,60,63,6
    2,63,43,40,40,56,55,13,1,24,8,40,44,42,46,17,42,62,53,37,17,41,50,27,
    46,53,55,63,46,55,23,59,31,59,63,24,63,36,28,36,33,9,9,10,45,23,53,55
    ,7,0
```

```
3990  DATA    24,8,24,8,40,5,24,8,41,22,41,9,9,50,17,18,30,23,17,23,23,31,
      19,59,27,7,24,28,31,27,7,24,24,8,12,1,24,8,24,8,24,8,5,40,21,18,26,18
      ,46,10,9,10,17,10,33,24,8,32,33,0
4000  REM  ***...SOUNDROUTINE ***
4010  DATA     173,48,192,136,208,5,206,1,3,240,9,202,208,245,174,0,3,76,2
      ,3,96
4020  REM *** ...MELODIE 1 ***
4030  DATA    63,64,111,64,103,64,111,64,127,64,134,64,127,64,134,64,147,6
      4,159,128,134,128,111,128,134,64,127,64,111,64,127,64,134,64,147,64,1
      27,128,103,128,111,128,103,128,111,255
4040  REM ***...MELODIE 2 ***
4050  DATA 228,100,228,100,228,100,171,220,114,220,128,100,136,100,152,100
      ,85,220,114,220,128,100,136,100,152,100,85,220,114,220,128,100,136,10
      0,128,100,152,255
4060  REM *** ...MELODIE 3 ***
4070  DATA  120,100,125,100,120,100,125,100,120,100,160,100,135,100,150,10
      0,180,255,240,100,180,100,160,255,240,100,160,100,150,255
4080  REM ***...KOORDINATEN DER SCHRIFTZUEGE ***
4090  DATA 9,5,8,5,6,2,8
4100  DATA 1,2,3,4,5,6,4,7
4110  REM "** (c) 1984 by **"
4120  REM "Norbert Faehrmann"
4130  STOP
```

Basic-Konverter



Das Programm besteht aus einer kleinen Assembler-Routine und zwei EXEC-Files. Wird das File mit CON.ABAS mit EXEC gestartet, verwandelt es ein APPLESOFT-Programm in ein INTERBASIC-Programm.

Das File CON.IBAS macht genau das Umgekehrte. Die Assembler-Routine stellt lediglich fest, in welchem Basic sich der Apple im Moment befindet und startet dann CON.ABAS bzw. CON.IBAS. Man muß also nur die Assembler-Routine CON.START entweder vom APPLESOFT mit CALL 768 eingeben, oder durch BRUN CON.START von der Diskette laden. Bei Speicherstelle $ 300 wird der Bildschirm gelöscht. Die Speicherstellen $ 0303 bis $ 0311 schreiben den Hello-Text. In den Speicherstellen $ 0312 bis $ 0320 wird das Byte in $ E000 untersucht. Enthält dieses den Wert $ 4C, dann befindet sich der Apple in APPLESOFT. Enthält das Byte $ 20, befindet er sich im INTERBASIC. Abhängig davon, wird nun nach ASTART oder ISTART verzweigt. ASTART schreibt dann EXEC CON.-ABAS oder ISTART schreibt EXEC CON.-ASTART.

Die beiden EXEC-Files sind im Grunde gleich. Wenn das File ausgeführt wird, fügt es zunächst die Zeile 0 an das momentane Programm an. Dann wird die Zeile 0 gestartet. Sie öffnet das File CON.TEXT und schreibt in dieses das Programm-Listing und den Befehl: DELETE CON.TEXT. Nun wird mittels INT auf INTERBASIC und das File CON.TEXT mit EXEC gestartet. Dadurch wird das Listing in den Speicher geholt, und zuletzt durch DELETE CON.TEXT auf der Diskette wieder gelöscht. Man befindet sich zum Schluß im INTEGERBASIC und kann mit LIST sein Programm auslisten. Anzumerken bleibt noch, daß in der Anweisung PRINT „EXEC CON.TEXT“ ein unsichtbares ctrl-D als erstes Zeichen steht.


```
10  REM  DIESES PROGRAMM ERSTELLT ALLE FILES
20  DATA 32,88,252,162,0,189,72,3,240,8,9,128,32,69,3,232,208,243,169,76
30  DATA 205,0,224,240,8,169,32,205,0,224,240,19,96,162,0,189,141,3,240
40  DATA 26,9,128,32,69,3,232,208,243,76,66,3,162,0,189,157,3,240,8,9
50  DATA 128,32,69,3,232,208,243,76,3,224,108,54,0,13,12,13,13,32,66,65
60  DATA 83,73,67,45,67,79,78,86,69,82,84,69,82,13,32,32,67,79,80,89,82
70  DATA 73,71,72,84,32,66,89,13,66,89,32,65,78,68,82,69,65,83,32,84,72
80  DATA 73,69,76,69,13,32,32,32,49,56,45,48,55,45,49,57,56,52,13,0,13,4
90  DATA 69,88,69,67,67,79,78,46,65,66,65,83,13,0,13,4,69,88,69,67,67,79
100  DATA 78,46,73,66,65,83,13,0
110  FOR I = 768 TO 940: READ X%: POKE I,X%: NEXT I
115  PRINT "SCHREIBE CON.START"
120  PRINT  CHR$ (4);"BSAVE CON.START,A$300,L$AD"
125  PRINT "SCHREIBE CON.ABAS"
130  PRINT  CHR$ (4);"OPEN CON.ABAS"
135  PRINT  CHR$ (4);"WRITE CON.ABAS"
140  PRINT "0?CHR$(4)"; CHR$ (34);"OPENCON.TEXT"; CHR$ (34);":?CHR$(4)"; CHR$
     (34);"WRITECON.TEXT"; CHR$ (34);":LIST1,:?"; CHR$ (34);"DELETECON.TEX
     T"; CHR$ (34);":?CHR$(4)"; CHR$ (34);"CLOSECON.TEXT"; CHR$ (34);":END
     "
145  PRINT "RUN"
150  PRINT "INT"
160  PRINT "PRINT"; CHR$ (34); CHR$ (4);"EXECCON.TEXT"; CHR$ (34)
165  PRINT  CHR$ (4);"CLOSE CON.ABAS"
170  PRINT "SCHREIBE CON.IBAS"
180  PRINT  CHR$ (4);"OPEN CON.IBAS"
185  PRINT  CHR$ (4);"WRITE CON.IBAS"
190  PRINT "0PRINT"; CHR$ (34); CHR$ (4);"OPENCON.TEXT"; CHR$ (34);":PRINT
     "; CHR$ (34); CHR$ (4);"WRITECON.TEXT"; CHR$ (34);":LIST1,32767:PRINT
     "; CHR$ (34);"DELETECON.TEXT"; CHR$ (34);":PRINT"; CHR$ (34); CHR$ (4
     );"CLOSECON.TEXT"; CHR$ (34);":END"
195  PRINT "RUN"
200  PRINT "FP"
210  PRINT "?CHR$(4)"; CHR$ (34);"EXECCON.TEXT"; CHR$ (34)
220  PRINT  CHR$ (4)"CLOSE CON.IBAS"
230  PRINT "FERTIG"
```

WILDCAT™

Looks Like an IBM™ Works Like an Apple™

To a land where fruit and flowers reign supreme, comes the awesome power of the **Wildcat** to challenge their supremacy.

Wildcat is a sleek styled mobile computer designed for your vehicle, boat or aircraft. It looks like an IBM PC™ and is fully software compatible with the Apple II™ product line at a list price that would put a smile on the face of the most frugal computer buyer.

But price isn't everything. All those features that would cost you hundreds of dollars extra from our competitors, come standard with **Wildcat.** Let's compare some of these features:



	Apple IIe	Wildcat
Detachable keyboard	No	STD
Full numeric key pad	Option	STD
Full functions keys	No	41
Built in disk controller	No	STD
Parallel printer port	No	STD
RS 232 serial port	No	STD
Game port	1	2
RGB video out	Option	STD
Composite video	STD	STD
RF video for TV	Option	STD
CP/M	Option	STD
Hi Res graphics (6 color)	STD	STD
Low Res graphics (16 color)	STD	STD
64KB memory	STD	STD
Half high disk drives	No	STD
Converters for vehicles, boats, and aircraft	No	Option
Aluminum carrying case	No	Option
List price	$1940*	$1099.00*

For more information on the all new **Wildcat**, see your local computer dealer or call or write:



COMPUTER AND PERIPHERAL PRODUCTS
1530 S. Sinclair
Anaheim, CA 92806
(714) 978-9820

*Computer plus one disk drive

Warning: This equipment is exempt from compliance with FCC testing requirements pursuant to 47 CFR 15.801 (c) (1)
Operation of this equipment in a residential area may cause interference.

IBM is the registered trademark of International Business Machines Corp.
Apple is the registered trademark of Apple Computer, Inc.
Apple II is the trademark of Apple Computer, Inc.

CIRCLE 181 ON READER SERVICE CARD

Fortsetzung von Seite 39

FORTH-Zahlensystem

In den letzten Teilen unseres Kurses haben wir uns ausschließlich mit dem ganzzahligen Zahlensystem beschäftigt. Wie bekannt, umfaßt dieses System den Zahlenbereich von −32768 bis +32767. Obwohl dieser Bereich nicht sehr umfangreich ist, reicht er doch für viele Anwendungen (z. B. Steuerungsaufgaben) völlig aus. Daß wir in *FORTH* auch mit längeren Zahlen arbeiten können, zeigt der folgende Abschnitt:

Doppelt lange Zahlen

Um in *FORTH* größere Zahlen zu ermöglichen, werden zwei normale Integer-Zahlen zu einer „Doppelten Zahl" zusammengefaßt. Durch diesen Trick steht jeder Zahl eine Matrix von vier Bytes zur Verfügung. Doppelt lange Zahlen umfassen somit den Zahlenbereich von −1.073741824 bis +1073741823.



Eine doppelt lange Zahl wird durch einen Dezimalpunkt gekennzeichnet. Dieser kann an jeder beliebigen Stelle sitzen und ist nicht mit dem üblichen Komma zu verwechseln, da dieser nur zur Kennzeichnung dient.

Beispiel:

Die gleiche Zahl verschieden geschrieben

1.234567
12.34567
123.4567
1234.567
12345.67
123456.7
1234567.

Da *FORTH* eigentlich für Steuerungsaufgaben entwickelt wurde und somit nur selten mit doppelt langen Zahlen gearbeitet wird, sind nur wenige Befehle vereinbart.

D. (d1 d2 d3 --› d1 d2)

Eine doppelt lange Zahl wird ausgegeben.

12345.67 D. ‹RETURN› 1234567 OK
1234.567 D. ‹RETURN› 1234567 OK

D+ (d1 d2 d3 --› d1 d)

Die Summe von d2 und d3 wird auf den Stack gelegt.

12000. 40000. D+ ‹RETURN› 52000 OK
90000. 5.000 D+ ‹RETURN› 95000 OK

D− (d1 d2 d3 --› d1 d)

Die Differenz von d2 und d3 wird auf den Stack gelegt.

50000. 10000. D− ‹RETURN› 40000 OK
55000. 44.000 D− ‹RETURN› 11000 OK

D.R (d1 d2 d3 n --› d1 d2)

Dieser Befehl entspricht weitgehend dem D.-Befehl. Druckt jedoch die Zahl rechtsbündig in ein Feld der Weite n. Das Feld (Fenster) n beginnt an der aktuellen Position des Ausgabegerätes. Einen ähnlichen Befehl kennen einige Basic-Interpreter unter der Bezeichnung ‚PRINT-USING'. Das folgende Beispiel zeigt die Leistungsfähigkeit dieses Befehles:

```
PRINT (6 Zahlen rechtsbündig ausgeben)
CR 7 D.R
CR 7 D.R
CR 7 D.R
CR 7 D.R
CR 7 D.R
CR 7 D.R
2. 20. 200. 2000. 20000. 200000.
‹RETURN› OK
```

```
PRINT ‹RETURN›
200000
 20000
  2000
   200
    20
     2 OK
```

DABS (d --› ud)

Der Absolutwert von d wird auf den Stack abgelegt.

−1234567. DABS ‹RETURN› 1234567 OK
−102. DABS ‹RETURN› 102 OK

DMAX (d1 d2 d3 --› d1 d)

Die Zahlen d2 und d3 werden verglichen und durch die größere Zahl ersetzt.

12345. 13400. DMAX D. ‹RETURN› 13400 OK
15000. 14000. DMAX D. ‹RETURN› 15000 OK

DMIN (d1 d2 d3 --› d1 d)

Die Zahlen d2 und d3 werden verglichen und durch die kleinere Zahl ersetzt.

14010. 12000. DMIN D.‹RETURN› 12000 OK
14050. 55000. DMIN D.‹RETURN› 14050 OK

Vergleichsbefehle für doppelte Zahlen

D= (d1 d2 d3 --› d1 f)
Flag wird 1, wenn d2 gleich d3 ist
D0= (d1 d2 d3 --› d1 d2 f)
Flag wird 1, wenn d3 gleich 0 ist
D‹ (d1 d2 d3 --› d1 f)
Flag wird 1, wenn d3 kleiner als d2 ist

Gemischte Zahlenoperationen

Bei der Multiplikation von Integer-Zahlen entstehen oft Überträge, die eine Integer-Zahl nicht mehr fassen kann. Es wäre deshalb wünschenswert, wenn es einen Befehl gäbe, welcher aus einer 16 Bit-Multiplikation ein 32 Bit-Ergebnis, also eine doppelt lange Zahl, bildet. Genau dieses gestattet der Befehl M★.

M★ (n1 n2 --› d1)

Aus dem Produkt zweier Integer-Zahlen wird eine doppelte Zahl gebildet.

Beispiele:

100 500 M★ D. ‹RETURN› 50000 OK
90 500 M★ D. ‹RETURN› 45000 OK
400 400 M★ D. ‹RETURN› 160000 OK

Das ganze funktioniert auch umgekehrt. Dividieren wir eine 32 Bit-Zahl durch eine 16 Bit-Zahl so erhalten wir eine 16 Bit-Integer-Zahl. Auch für diesen Fall gibt es einen Befehl: M/

M/ (d1 n1 --› n1)

Eine doppelte Zahl dividiert durch eine Integer-Zahl ergibt eine Integer-Zahl.

Beispiele:

50000. 20 M/. ‹RETURN› 2500 OK
100000. 50 M/. ‹RETURN› 2000 OK
50000. 25000 M/ . ‹RETURN› 2 OK

Zahlen ohne Vorzeichen

Normalerweise verwendet *FORTH* zum Abspeichern von Integer-Zahlen immer nur 15 Bit. Das 16. Bit kennzeichnet, welches Vorzeichen vorhanden ist, Plus oder Minus. Eine Integer-Zahl kann somit den Bereich von −32768 bis 32767 umfassen. Verzichtet man auf das Vorzeichen, so würde der Zahlenbereich von 0 bis 65535 gehen. Diese Möglichkeit bietet der Befehl U.

U. (n1 n2 n3 --› n1 n2)

Die Zahl n3 wird ohne vorzeichen ausgegeben.

```
1000 U.   ‹RETURN›  1000 OK
30000 U.  ‹RETURN› 30000 OK
60001 U.  ‹RETURN› 60001 OK
   -1 U.  ‹RETURN› 65535 OK
-1000 U.  ‹RETURN› 64536 OK
```

Die normale Ausgabe würde wie folgt aussehen:

```
   -1 .   ‹RETURN› -    1 OK
-1000 .   ‹RETURN› -1000 OK
```

Operationen ohne Vorzeichen

U★ (u1 u2 --› ud)
Zwei vorzeichenlose Zahlen werden miteinander multipliziert. Das Ergebnis wird ebenfalls als vorzeichenlose doppelte Zahl abgelegt.

40000 44000 U★ D. ‹RETURN›
1760000000 OK
30000 34500 U★ D. ‹RETURN›
1035000000 OK
U/MOD (ud u1 --› u2 u3)

Eine doppelte Zahl wird durch eine einfache dividiert. Das Ergebnis und der Rest werden als einfache Zahl auf den Stack gelegt.

U/ (u1 u2 --› f)
Flag wird 1, wenn u1 kleiner als u2 ist.
100 1000 U/ . ‹RETURN› 1 OK
1000 100 U/ . ‹RETURN› 0 OK

FORTH bietet auch die Möglichkeit eine DO-LOOP-Schleife ohne Vorzeichen auszuführen.

DO (u1 u2 --› empty)
u1 gleich Endwert der Schleife und u2 gleich Anfangswert der Schleife.
LOOP (u --› empty)
u gleich Schrittweite der Schleife.

Beispiel:

```
Test 50000 0 DO
     CR
     I U.
     50001 /LOOP
TEST ‹RETURN›
0
5001
10002
15003
20004
25005
30006
35007
40008
45009 OK
```

Gleitkommazahlen

Bis jetzt haben wir viele Probleme ohne Gleitkommazahlen gelöst. Es ist also offensichtlich, daß man sehr oft auf die Rechnung in Gleitkommadarstellung verzichten kann. Gleichzeitig verzichtet man dabei auf die auftretenden Rundungsfehler, von denen Basic-Programmierer „ein Lied singen können". In den wenigen Fällen, wo man auf eine Berechnung in Gleitkommadarstellung nicht verzichten kann, wird man ein Gleitkommapaket, das zu vielen *FORTH-Versionen* erhältlich ist, hinzuladen.

Auf eine Darstellung dieser Befehlsworte kann in diesem Kurs nicht eingegangen werden, da es keine Standarts gibt und somit die Befehle stark von einander abweichen.

Diese sind jedoch in dem jeweiligen Manual ausführlich beschrieben.

Gesamtverzeichnis aller FORTH-Befehle: Wir haben nun eine Menge über die Sprache *FORTH* gelernt. Ziel dieses Kurses war es, dem Einsteiger einen Einblick und erste Programmierkenntnisse zu vermitteln. Für diejenigen unter Ihnen, die tiefer in die neue Sprache eindringen möchten, empfehlen wir am Ende noch einige interessante Literatur.

Zum Abschluß unseres *FORTH*-Kurses bringen wir noch einmal eine alphabetische Auflistung der wichtigsten Forth-Befehle und deren Aufgabe. Je nach *FORTH*-Version können mehr oder weniger Befehle implementiert sein. Dies ist aus dem jeweiligen Manual zu entnehmen.

! (n adr --> empty)
Speichere die 16-Bit-Zahl ›n‹ in die Adresse ›adr‹. Vergleichbar mit Poke in Basic, jedoch wird hier eine 16-Bit-Zahl gepokt.

!CSP (adr n --> empty)
Der aktuelle StackPointer wird in die User-Variable CSP gerettet.

\# (ud1 --> ud2)
Wandlung einer Zahlenstelle.

\#> (ud --> adr n)
Ende der Formatierung. Auf dem Stapel bleibt die Adresse der Zeichenkette und die Anzahl der Zeichen.

\#S (ud1 --> ud2)
Alle verbleibenden Stellen werden in ASCII Zeichen gewandelt.

' (empty --> adr)
Die Parameter-Feldadresse eines Befehlswortes oder einer Variablen wird auf dem Stack abgelegt.

< (empty --> empty)
Kommentarbeginn. Vergleichbar mit dem Basic-Befehl REM.

☆ (n1 n2 --> n3)
Das Produkt der Integer-Zahlen n1 und n2 wird auf dem Stack abgelegt.

☆/ (n1 n2 n3 --> n4)
Der Verhältnis-Faktor n4 wird aus der Formel n4 = n1☆n2/n3 errechnet. Das Produkt n1☆n2 wird doppelt so lang berechnet.

☆/MOD (n1 n2 n3 --> n4 n5)
Das gleiche wie der ☆/-Befehl. Jedoch wird auch hier der Rest n5 auf den Stack gelegt.

\+ (n1 n2 --> n3)
Plus-Befehl! Die Summe aus den Zahlen n1 und n2 wird auf den Stack gelegt.

+! (n adr --> empty)
Die Zahl in Speicherstelle 1 wird um 1 erhöht.

+- (n1 n2 --> n3)
Das Vorzeichen von n2 wird auf die Zahl n1 angewandt. Das Ergebnis n3 wird auf dem Stack abgelegt.

+LOOP (n1 --> empty)
Das Zahlenparameter der Schleife wird um den Wert n1 geändert. Der vergleichbare Basic-Befehl wäre NEXT in Verbindung mit STEP.

\- (n1 n2 --> n3)
Minus-Befehl! Die Differenz aus den Zahlen n1 und n2 wird auf dem Stack abgelegt.

--> (empty --> empty)
Dieser Befehl wird zum Verbinden mehrerer Screens genutzt. Er wird in der Regel auf die letzte Zeile des Screens gesetzt und bewirkt dadurch, daß der Compiler mit dem nächsten Screen fortfährt.

-DUP (n1 --> n1) falls n1 = 0
(n1 --> n1 n1) falls n1 <> 0
Die Zahl n1 wird mit 0 verglichen. Ist diese ungleich, wird sie dupliziert.

-TRAILING (adr n1 --> adr n2)
Ändert für die Ausgabe von Text die Parameter. Das führende Leerzeichen wird übersprungen.

. (n --> empty)
Die 16-Bit-Zahl wird auf dem Schirm ausgegeben. Beachtet wird die gewählte Zahlen-Basis.

." Beispiel: ." xxxxx"
Der Text xxxxx wird auf dem Schirm ausgegeben. Der vergleichbare Befehl wäre PRINT.

.R (n1 n2 --> empty)
Die Zahl n1 wird rechtsbündig in einem Fenster der Breite n2 ausgegeben. Der vergleichbare Basic-Befehl PRINT-USING.

/ (n1 n2 --> n3)
Divisions-Befehl! Der Quotient aus den Zahlen n1/n2 wird auf dem Stack abgelegt.

/MOD (n1 n2 --> n3 n4)
Das gleiche wie /-Befehl. Jedoch wird auch hier der Rest n3 auf dem Stack abgelegt.

Da diese Zahlen in *Forth* häu-

fig benutzt werden, sind diese Zahlen auch als Befehl definiert.

0< (n --> Flag)
Vergleichsbefehl! Ist die Zahl kleiner als 0, also im negativen Bereich, so wird das Flag auf 1 ansonsten auf 0 gesetzt.

0= (n --> Flag)
Vergleichsbefehl! Ist die Zahl gleich 0 so wird das Flag auf 1 ansonsten auf 0 gesetzt.

: (empty --> empty)
Beginn der Definition eines neuen Befehles.

; (empty --> empty)
Dies ist das Gegenstück zum >":"< Befehl. Das Ende einer Definition wird gekennzeichnet.

;S (empty --> empty)
Ende der Compilierung. Findet der *FORTH*-Compiler diesen Befehl vor, so beendet er den Compiler-Vorgang rechtzeitig.

< (n1 n2 --> Flag)
Vergleichsbefehl! Ist die Zahl n1 kleiner als n2, wird das Flag auf 1 ansonsten auf 0 gesetzt.

>R (n1 --> empty)
Die Zahl n wird zum Return-Stack gebracht. Mit diesem Befehl sollte sehr vorsichtig umgegangen werden, da es sonst leicht zum Absturz des Systems führen kann.

? (adr --> empty)
Der Inhalt der Adresse adr wird auf dem Schirm ausgegeben.

?TERMINAL (empty --> f)
Die Tastatur wird abgefragt. Ist eine Taste gedrückt, so wird das Flag auf 1 ansonsten auf 0 gesetzt.

□ (adr --> n)
Der Inhalt der Adresse adr wird auf den Stack gelegt. Der vergleichbare Basic-Befehl wäre PEEK, jedoch wird hier eine 16 Bit-Zahl gelesen.

ABS (n --> u)
Der Absolutwert von n wird auf dem Stack abgelegt. Mit anderen Worten, das Vorzeichen wird abgeschnitten.

ALLOT (n --> empty)
)n(Bytes Platz wird im Wörterbuch reserviert. Der Dictionary-Pointer DP wird also um n Bytes erhöht.

AND (n1 n2 --> n3)
Die Zahlen n1 und n2 werden logisch UND-verknüpft. Zu beachten ist, daß diese Verknüpfung bitweise vorgenommen wird.

BASE (n --> n adr)
Diese Variable enthält die gegenwärtig vereinbarte Zahlenbasis.

BLOCK (n --> adr)
Die Speicheradresse welche den Block n enthält wird ermittelt und auf den Stack gelegt.

C! (b adr --> empty)
Die-8 Bit-Zahl b wird in Adresse adr übertragen. Der vergleichbare Basic-Befehl wäre POKE.

C, (b --> empty)
Die-8 Bit-Zahl b wird in den nächsten freien Speicherplatz gebracht. Der Dictionary-Pointer wird um 1 erhöht.

C□ (adr --> b)
Die Zahl in Adresse adr wird als 8-Bit-Zahl auf dem Stack abgelegt. Das High-Byte wird also auf 0 gesetzt. Der vergleichbare Basic-Befehl wäre PEEK.

CFA (pfa --> cfa)
Die Parameterfeldadresse wird in die Codefeldadressé umgerechnet.

COMPILE
Das folgende Wort wird während der Laufzeit in das Wörterbuch eingetragen.

CMOVE (adr adr2 n --> empty)
Verschiebt n Byte von Adresse adr1 nach Adresse adr2.

CONSTANT (n --> empty)
Eine Konstante mit dem Wert n wird angelegt. Programme, welche zahlreiche Konstanten enthalten, sind in der Regel etwas schneller.

COPY (n1 n2 --> empty)
Das Textfeld n1 wird nach Textfeld n2 übertragen.

CONTEXT (empty --> adr)
Diese User-Variable legt die Anfangsadresse des aktuellen Wörterbuches auf den Stack.

COUNT(adr1 adr2 n)
Auf den Stapel wird die Länge und die Anfangsadresse eines Textes abgelegt.

CR
(empty --> empty)
Dieser Befehl bewirkt einen Zeilenvorschub mit Wagenrücklauf.

CREATE (empty --> empty)
Ist eines der wichtigsten Definitionsworte des FORTH-Kernes. Ein Wort wird in das Wörterbuch eingetragen und die Codefeldadresse wird Parameterfeldadresse des Eintrags.

CURRENT (empty --> adr)
Diese Variable enthält die Adresse des Wörterbuches, in die nachträgliche Definitionen eingetragen werden.

D+ (d1 d2 --> d3)
Zwei doppelt genaue Zahlen werden addiert und das Ergebnis auf dem Stack abgelegt.

D– (d1 d2 --> d3)
Die Differenz aus den beiden doppelt genauen Zahlen wird auf dem Stack abgelegt.

D. (d --> empty)
Eine doppelt genaue Zahl wird auf dem Schirm ausgegeben.

D.R (d n --> empty)
Die doppelt genaue Zahl d wird rechtsbündig in einem n breiten Fenster ausgegeben.

D0= (d --> f)
Vergleichsbefehl! Wenn die doppelt genaue Zahl gleich 0 ist, wird das Flag auf 1 gesetzt. Ansonsten auf 0.

D< (d1 d2 --> f)
Vergleichsbefehl! Wenn die doppelt genaue Zahl d2 kleiner als d1 ist, wird das Flag auf 1, ansonsten auf 0 gesetzt.

DECIMAL (empty --> empty)
Die dezimale Betriebsart wird aktiviert.

DO (n1 n2 --> empty)
Eine Schleife wird eröffnet mit dem Anfangsparameter n2 und dem Endparameter n1. Der vergleichbare Basic-Befehl wäre FOR.

DOES> (empty --> empty)
Dieses Wort wird in Definitionsworten verwendet, es beendet den Teil für die Compile-Anweisungen und zeigt den Beginn der Laufzeitanweisungen an.

DROP (n --> empty)
Die oberste Zahl des Stacks wird gelöscht.

DUP (n1 --> n1 n1)
Die oberste Zahl des Stack's wird dupliziert.

EMIT (c --> empty)
Die oberste Zahl des Stack's wird als ASCII-Zeichen auf dem Schirm ausgegeben.

ERASE (adr n --> empty)
Der Speicher wird ab Adresse adr mit n Bytes des Wertes 0 gefüllt.

EXECUTE (adr --> empty)
Ein Programm, welches bei der Adresse adr beginnt, wird zur Ausführung gebracht. Die Adresse entspricht der Codefeldadresse.

EXPECT (adr n --> empty)
Eine Anzahl von n Zeichen können an Adresse adr eingegeben werden.

FENCE (empty --> adr)
Diese Variable enthält die Adresse des ersten ungeschützten Wörterbucheintrages.

FILL (adr n c --> empty)
Der Speicher ab Adresse adr wird mit n Bytes des Wertes c gefüllt.

FLUSH (empty --> empty)
Das zuletzt bearbeitete Textfenster wird abgespeichert.

FORGET cccc (empty --> empty)
Alle Definitionen ab dem Wort cccc werden gelöscht.

FORTH (empty --> empty)
Das Wörterbuch mit der Bezeichnung *FORTH* wird aktiviert.

HERE (empty --> adr)
Dieser Befehl bringt die Adresse des nächsten freien Speicherplatzes auf den Stack.

HEX (empty --> empty)

Die Hexadezimale Betriebsart wird aktiviert.
HOLD (c --> empty)
Dieser Befehl erlaubt das Einfügen eines ASCII-Zeichens in eine Zeichenkette.
I (empty --> n)
Dieser Befehl wird innerhalb einer Schleife genutzt, um das aktuelle Zahlenparameter auf den Stack zu bringen.
IMMEDIATE empty --> empty)
Eine Definition wird als unmittelbar erklärt und sofort ausgeführt.
KEY (empty --> c)
Es wird ein Zeichen von der Tastatur eingelesen und als ASCII-Wert auf den Stack gelegt.
LATEST (empty --> adr)
Diese Variable enthält die Namensfeldadresse der letzten Definition.
LEAVE (empty --> empty)
Durch diesen Befehl kann eine DO-...LOOP Schleife verlassen werden.
LFA (pfa --> lfa)
Die Parameterfeldadresse wird in die Verbindungsfeldadresse umgerechnet.
LIST (n --> empty)
Das Textfenster n wird auf dem Schirm ausgelistet.
LITERIAL (n --> empty)
Bei der Compilierung von n wird vom Stack in die Wortdefinition eingefügt.
LORD (n --> empty)
Der Compiler beginnt bei Screen n zu compilieren.
M☆ (n1 n2 --> d1)
Die Integer-Zahlen n1 und n2 werden miteinander multipliziert und das Ergebnis als 32 Bit-Zahl auf den Stack gelegt.
M/ (d n1 --> n2)
Die doppelt genaue Zahl d wird durch Zahl n1 definiert und das Ergebnis auf den Stack gelegt.
MAX (n1 n2 --> n)
Die größere der beiden obersten Zahlen bleibt auf dem Stack.
MESSAGE (n --> empty)
Die Meldung n wird auf dem Schirm ausgegeben. Dieser Befehl wird vom *FORTH*-System zum Ausgeben der Fehlermeldungen verwendet.
MIN (n1 n2 --> n)
Die kleinere der beiden obersten Zahlen bleibt auf dem Stack.
MINUS (n --> -n)
Die oberste Zahl des Stacks wird negativ.
MOD (n1 n2 --> n)
Der Rest der Division n1 durch n2 wird auf dem Stack abgelegt.
NFA (pfa --> nfa)
Die Parameterfeldadresse wird in die Namensfeldadresse umgerechnet.
NUMMER (adr --> d)
Eine Zeichenkette, die bei Adresse adr+1 beginnt, wird in eine doppelt lange Zahl gewandelt. Die Adresse adr zeigt dabei auf die Länge der Zeichenkette.
OR (n1 n2 --> n)
Die Zahlen n1 und n2 werden miteinander durch OR verknüpft.
OVER (n1 n2 --> n1)
Die vorletzte Zahl des Stacks wird dupliziert und auf den Stack gelegt.
PAD (empty --> adr)
Die Adresse des Speichers namens PAD wird auf den Stack gelegt.
PFA (nfa --> pfa)
Die Namensfeldadresse wird in die Parameterfeldadresse umgerechnet.
QUIT (empty --> empty)
Die Laufzeit wird beendet ohne die Ausgabe von OK.
R (empty --> n)
Die oberste Zahl des RETURN-Stacks wird auf den Rechen-Stack gebracht, ohne sie auf dem RETURN-Stack zu zerstören.
R> (empty --> n)
Die oberste Zahl des RETURN-Stacks wird auf den Rechen-Stack geholt.
ROT (n1 n2 n3 --> n2 n3 n1)
Die obersten 3 Zahlen des Stacks werden um eine Stelle rotiert.
S0 (emply --> adr)
Diese Variable enthält die Adresse des Stackendes.
SCR (empty --> adr)
In dieser Variablen wird der zuletzt bearbeitete Screen festgehalten.
SP□ (empty --> adr)
Dieser Befehl legt die Adresse des augenblicklichen TOS auf den Stack. Der Befehl selbst wird von dieser Berechnung ausgeschlossen.
SPACE (empty --> empty)
Ein SPACE-Zeichen (dezimal 32) wird auf dem Schirm ausgegeben.
SPACES (n --> empty)
)n(SPACE-Zeichen werden auf dem Schirm ausgegeben.
SWAP (n1 n2 --> n2 n1)
Die beiden obersten Zahlen des Stacks werden vertauscht.
TIP (empty --> adr)
Diese Variable enthält die Startadresse des Terminal-Input-Puffers.
TYPE (adr n --> empty)
Eine Zeichenkette, deren Länge n ist und ab adr gespeichert wird, erscheint auf dem Schirm.
U☆ (u1 u2 --> ud)
Die beiden vorzeichenlosen 16 Bit-Zahlen werden miteinander multipliziert und das Ergebnis als doppelt genaue Zahl auf den Stack abgelegt.
U/ (ud u1 --> u2 u3)
Der Quotient der beiden Zahlen ud und u1 wird unter u3 und der Rest unter u2 auf den Stack gelegt.
UPDATE (empty --> empty)
Der zuletzt bearbeitete Screen wird als geändert markiert.
USER (name) (n --> empty)
Dieser Befehl trägt eine Variable in einen vom Benutzer definierten Bereich ein.
VARIABLE (name) (n --> empty)
Eine Variable (name) wird angelegt und mit dem Wert n geladen.
VOCABULARY (name) (empty --> empty)
Mit diesem Befehl läßt sich ein neues Wörterbuch eröffnen.
VLIST (empty --> empty)
Das eingeschaltete Wörterbuch wird auf dem Schirm ausgelistet.
WORD (c --> empty)
Eine Zeichenfolge wird aus dem Eingabespeicher (TIB) bis zum Begrenzungszeichen c übernommen.
XOR (n1 n2 --> n)
Die Zahlen n1 und n2 werden EXCLUSIV-ODER verknüpft und das Ergebnis auf dem Stack abgelegt.
[(empty --> empty)
Dieses Zeichen schaltet den Compiler aus.
] (empty --> empty)
Dieses Zeichen schaltet den Compiler ein.

Weiterführende Literatur:

STARING FORTH; Leo Brodie; englisch
Dieses Werk ist sicher eines der besten FORTH-Bücher, die es auf dem Markt gibt. Mit viel Humor und witzigen Skizzen wird man in die etwas schwierige Sprache Forths eingeführt. Leider gibt es bis jetzt noch keine deutsche Übersetzung.

FORTH HANDBUCH; E. Flögel; deutsch
Ein Buch, das zu den wenigen deutschen Büchern zählt, welche sich mit Forth beschäftigen. Der Leser wird Schritt für Schritt in den umfangreichen Wortschatz von FORTH eingeführt, dabei wird in erster Linie der FIG-Forth Standard besprochen. Einige kurze Programme runden dieses Werk ab.

DIE PROGRAMMIERSPRACHE FORTH; Ronald Zech; deutsch
In diesem Buch wird die Programmiersprache FORTH streng methodisch und umfassend beschrieben. Darüber hinaus wird Fundamentales erläutert, auf das die Sprache aufgebaut ist.
Da dieses Werk leider mit sehr vielen Fachwörtern umgeht, ist es nicht dem Anfänger zu empfehlen. Frank Brall

Topprogramm

Ausgezeichnet von der Redaktion als Top-Programm in „Computronic". Bewertet wurden Spielidee, Bewegungsablauf und Grafik.



Sie steigen in den Alltag einer Flaschenfabrik ein. Dort müssen Sie Bierkisten zusammenstellen. Die Arbeit wird Ihnen aber durch zahlreiche Hindernisse erschwert.

Nach dem Start mit „RUN" erklingt eine Melodie. Danach werden die Bildnummer und die Schwierigkeitsphase angezeigt. Eine Phase besteht aus vier Bildern (4 Aufgaben). Nach der Anzeige, die nach jedem geschafften Bild eingeblendet wird, erscheint das Bild Nr. 1. Oben links wird die Anzahl der verfügbaren Leben angezeigt, d. h., am Anfang hat jeder Spieler drei Leben. Links unten erscheint das erste Männchen, welches mit dem Joystick gesteuert wird. Der Bildschirmaufbau besteht aus mehreren Etagen, die durch Leitern verbunden sind. Auf den Etagen befinden sich Fließbänder, die Flaschen transportieren. Die Anzahl der Fließbänder nimmt bis zum vierten Bild zu, und ab Bild Nr. 2 werden diese teilweise unsichtbar. Außerdem befinden sich 9 Bierfässerteilchen auf dem Bildschirm, die für den Spieler sehr wichtig sind, denn sie haben beim Start eines Bildes ihre bestimmte Anfangsposition, die bei den vier Bildern jeweils unterschiedlich ist. Durch Berühren eines Fässerteilchens mit dem Männchen, verändert sich die Position dieses Teilchens und die direkt unter ihm liegenden Teilchen. Wie und was durch so eine Berührung passiert, ist aus den Beispielen 1 und 2 zu entnehmen. Hat man alle Bierfässerteile (durch Berührung) nach unten transportiert, so erkennt man 3 Bierfässer. Wenn man dieses erreicht hat, ist das Bild bzw. die Aufgabe erfüllt, und es erscheint die Anzeige der Phase und Bildschirmnummer. Danach folgt das nächste Bild. Dieses hört sich alles relativ einfach an, doch es gibt einen Schwachpunkt des Männchens. Es verträgt keinen Alkohol. Deswegen darf es keine auf dem Bildschirm befindlichen Flaschen berühren, da sie Alkohol enthalten. Dieses kann durch einen Sprung verhindert werden (die Steuerung des Männchens ist aus dem Beispiel 3 zu entnehmen). Eine weitere Gefahr ist die Zeit. Oben am Bildschirm erkennt man eine Etage, auf dem das Männchen des Spielers symbolisiert wird. Von rechts her nähert sich eine Whiskyflasche. Sollte diese zu dem symbolisierten Männchen gelangen, so hat der Spieler ein Leben verloren. Da dieses sehr schnell geht, hat der Spieler noch eine Chance, durch Berühren des Malzbieres, die Flasche wieder von vorn (also von rechts) starten zu lassen. Das Malzbier erscheint an einer bestimmten Position. Wann es erscheint, hängt von dem Berühren eines der drei Oberteile der Bierfässer ab. Hierzu ein Beispiel: das Malzbier ist nicht auf dem Bild, aber wenn man jetzt eines der Oberteile berührt, dann erscheint es. Bei nochmaliger Berührung verschwindet es und bei der nächsten Berührung erscheint es wieder. Also ein bestimmter Rhythmus, das Malzbier muß eindeutig berührt werden.

Punkte werden beim Versetzen eines Faßteiles gegeben oder nach Beendigung eines Bildes. Die Entfernung der Whiskyflasche zum Männchen ist für die Höhe der Punkte ausschlaggebend. Die aktuelle Punktzahl wird nach Verlust eines Männchens oder bei anderen Unterbrechungen angezeigt, aber nicht während des Spieles. Sollte man eine Phase geschafft haben, so erhält man ein zusätzliches Leben. Sollte man alle Leben verbraucht bzw. verspielt haben, so erscheint „GAME-OVER".

Durch Betätigung einer Taste, beginnt das Spiel von vorn. Ansonsten ist das Programm musikalisch untermalt.

```
10 CALL SCREEN(2):: FOR I=3 TO 8 :: CALL COLOR(I,9,1):: NEXT I
20 CALL CLEAR
30 DISPLAY AT(9,9):"ALKOHOLVERBOT" :: DISPLAY AT(11,14):"VON" :: DISPLAY AT(13,1
0):"JENS BARTHE"
40 CALL MUSIK
50 CALL KEY(1,K,S):: IF K=18 THEN 60 ELSE CALL KEY(0,K,S):: IF S=0 THEN 50
60 CALL CLEAR :: CALL MAGNIFY(3)
70 CALL CHAR(95,"1C3E1C083E1C1C14",97,"0103000700070F0F",98,"FFFF00FF00FFFFFF",9
```



```
9,"80C000E000E0F0F0")
80 CALL CHAR(100,"0F0F070700030001",101,"FFFFFFFF00FF00FF",102,"F0F0E0E000C0008"
,113,"FFFFFFFF")
90 CALL CHAR(104,"0F0F080A080A080F",105,"FFFFA2AEA2AEA2FF",106,"F0F030B07070B0F"
,113,"FFFFFFFF")
100 CALL CHAR(120,"192640405F3F5F5B",121,"B8440204F8F0FEBF",122,"395A5B9B7B5B1F1
F",123,"33B3B3BEBCB0F0F0",124,"FF81FF81FF81FF81")
110 CALL CHAR(36,"01030F02070301010B0703030101010380C0C0E0E0C080008080808000000
00")
120 CALL CHAR(40,"01030307070303010303030301010101 80C0F040E0C00000A0C08080000000
80")
130 CALL CHAR(60,"03070F0F0F0713130F0707070404040C0080C0C0C0802020C0808080808080
C0")
140 CALL CHAR(108,"03020301010101010608304D4D26100FC040C08080808080 60100CB2B2640
8F0")
150 CALL CHAR(116,"01030F02070301010B07030303060818 80C0C0E0E0C08000C0A080808080E
020")
160 CALL CHAR(128,"010303070703030103070B0303020E0880C0F040E0C00000A0C0808080C02
030")
170 CALL CHAR(132,"0000000000000000444444EEAAAAEE000000000000000000444444EEAAA
AEE")
180 CALL CHAR(136,"000307030202020204050508090807 00C0E0C04040404040 20A02090901
0E0")
190 CALL CHAR(140,"000301010101010102040909080403 00E0C0C040404040 4020D00808C81
0E0")
200 IF SC>HS THEN HS=SC
210 M=2 :: BI,SC=0 :: PH=1
220 BI=BI+1 :: IF BI>4 THEN 230 ELSE 240
230 M=M+1 :: PH=PH+1 :: BI=1
240 CALL CLEAR :: CALL DELSPRITE(ALL):: DISPLAY AT(11,11):"BILD :";BI :: DISPLAY
 AT(13,10):"PHASE :";PH :: FOR I=1 TO 300 :: NEXT I
250 Q1=INT(-2-PH/2):: Q2=INT(4+PH/3):: Q3=INT(-7-PH):: Q4=INT(-3-PH/2):: Q5=INT(
5+PH/3):: Q6=INT(3+PH/3):: Q7=INT(2+PH/3)
260 CALL CLEAR :: ON BI GOSUB 320,410,540,670
270 DISPLAY AT(24,1):"HI-SCORE:";HS :: DISPLAY AT(24,17):"SCORE:";SC
280 CALL HCHAR(2,2,95,M)
290 CALL SPRITE(#20,128,16,1,40,#21,108,13,1,256,0,-1):: CALL SPRITE(#1,40,16,15
3,6):: CALL HCHAR(3,6,113,27)
300 GOTO 810
310 ! BILD 1
320 CALL COLOR(9,11,1,10,11,1,11,16,1,12,5,1)
330 CALL HCHAR(1,1,32,8):: A(1),B(1),C(1)=8 :: A(2),B(2),C(2)=13 :: A(3),B(3),C(
3)=18 :: AA=7 :: BB=17 :: CC=27
340 CALL HCHAR(9,1,113,32):: CALL HCHAR(14,1,113,32):: CALL HCHAR(19,1,113,32)::
 CALL HCHAR(22,1,113,2)
350 CALL VCHAR(9,2,124,5):: CALL VCHAR(14,12,124,5):: CALL VCHAR(14,22,124,5)::
CALL VCHAR(9,31,124,5)
360 CALL VCHAR(14,3,124,5):: CALL VCHAR(19,2,124,3):: CALL VCHAR(14,32,124,5)
370 CALL SPRITE(#2,140,14,129,60,#3,140,14,129,188,#4,136,4,89,80,#5,136,4,89,20
8)
380 CALL SPRITE(#6,132,7,49,40,#7,132,7,49,168)
390 CALL MOTION(#2,0,Q1,#3,0,Q1):: CALL MOTION(#4,0,Q2,#5,0,Q2):: CALL MOTION(#6
,0,Q3,#7,0,Q3):: RETURN
400 ! BILD 2
410 CALL COLOR(11,8,1,12,5,1)
420 A(1)=6 :: C(1)=5 :: A(2)=11 :: C(2)=10 :: A(3)=16 :: C(3)=15
430 CALL HCHAR(1,1,32,8):: CALL HCHAR(7,2,113,9):: CALL HCHAR(6,14,113,7):: CALL
 HCHAR(6,26,113,7):: CALL HCHAR(8,20,113,5)
440 CALL HCHAR(9,16,113,4):: CALL HCHAR(11,11,113,4):: CALL HCHAR(11,24,113,9)::
 CALL HCHAR(12,4,113,6)
450 CALL HCHAR(14,1,113,22):: CALL HCHAR(16,25,113,4):: CALL HCHAR(16,31,113,2):
: CALL HCHAR(17,6,113,4)
460 CALL HCHAR(19,2,113,5):: CALL HCHAR(19,9,113,23):: CALL HCHAR(22,1,113,3)::
```

```
CALL HCHAR(4,2,113,3)
470 CALL VCHAR(4,3,124,3):: CALL VCHAR(7,2,124,12):: CALL VCHAR(7,4,124,5):: CAL
L VCHAR(7,11,124,4):: CALL VCHAR(6,14,124,5)
480 CALL VCHAR(6,19,124,3):: CALL VCHAR(6,20,124,2):: CALL VCHAR(8,24,124,3):: C
ALL VCHAR(12,9,124,2):: CALL VCHAR(11,32,124,5)
490 CALL VCHAR(14,21,124,5):: CALL VCHAR(16,31,124,3):: CALL VCHAR(17,6,124,2)::
 CALL VCHAR(17,9,124,2):: CALL VCHAR(19,3,124,3)
500 CALL VCHAR(6,31,124,5):: CALL SPRITE(#2,140,14,129,60,#3,140,14,129,188,#4,1
36,4,89,80,#5,136,4,89,208)
510 CALL SPRITE(#6,132,7,33,70,#7,132,7,65,198):: CALL MOTION(#6,0,PH+5,#7,0,Q3)
520 CALL MOTION(#2,0,Q4,#3,0,Q4,#4,0,Q5,#5,0,Q5):: RETURN
530 ! BILD 3
540 CALL COLOR(11,5,1,12,14,1)
550 C(1)=6 :: C(2)=11 :: C(3)=16
560 CALL HCHAR(1,1,32,8):: CALL HCHAR(7,1,113,9):: CALL HCHAR(7,13,113):: CALL H
CHAR(6,21,113):: CALL HCHAR(8,22,113)
570 CALL HCHAR(9,16,113,5):: CALL HCHAR(12,4,113,6):: CALL HCHAR(12,14,113):: CA
LL HCHAR(14,15,113,7):: CALL HCHAR(12,26,113,3)
580 CALL HCHAR(17,1,113,8):: CALL HCHAR(17,26,113,7):: CALL HCHAR(19,9,113,4)::
CALL HCHAR(19,16,113,3):: CALL HCHAR(19,22,113,4)
590 CALL HCHAR(14,25,113):: CALL HCHAR(22,1,113,3):: CALL VCHAR(7,2,124,10):: CA
LL VCHAR(17,3,124,5):: CALL VCHAR(12,4,124,5)
600 CALL VCHAR(7,9,124,5):: CALL VCHAR(17,9,124,2):: CALL VCHAR(7,14,124,5):: CA
LL VCHAR(12,15,124,2):: CALL VCHAR(6,20,124,3)
610 CALL VCHAR(9,21,124,3):: CALL VCHAR(6,22,124,2):: CALL VCHAR(12,25,124,2)::
CALL VCHAR(17,25,124,2):: CALL VCHAR(12,29,124,5)
620 CALL HCHAR(7,26,113,7):: CALL HCHAR(12,21,113,2):: CALL VCHAR(7,31,124,10)
630 CALL SPRITE(#2,140,15,129,60,#3,140,15,113,188,#4,136,4,73,80,#5,132,7,33,19
8)
640 CALL SPRITE(#6,136,4,89,208,#7,132,7,49,70)
650 CALL MOTION(#2,0,Q1,#3,0,Q6,#4,0,Q2,#5,0,Q7):: CALL MOTION(#6,0,Q1,#7,0,Q4):
: RETURN
660 ! BILD 4
670 CALL COLOR(11,13,1,12,7,1)
680 A(1)=7 :: B(1)=5 :: C(1)=4 :: A(2)=12 :: B(2)=10 :: C(2)=9 :: A(3)=17 :: B(3
)=15 :: C(3)=14 :: CC=26
690 CALL HCHAR(1,1,32,8):: CALL HCHAR(5,24,113,4):: CALL HCHAR(5,32,113):: CALL
HCHAR(6,16,113,3):: CALL HCHAR(8,4,113,12)
700 CALL HCHAR(8,19,113):: CALL HCHAR(8,22,113,2):: CALL HCHAR(10,24,113,4):: CA
LL HCHAR(10,31,113):: CALL HCHAR(11,12,113)
710 CALL HCHAR(13,4,113,7):: CALL HCHAR(14,18,113,6):: CALL HCHAR(15,11,113):: C
ALL HCHAR(15,25,113,3):: CALL HCHAR(16,17,113)
720 CALL HCHAR(18,1,113,3):: CALL HCHAR(18,6,113,3):: CALL HCHAR(18,12,113):: CA
LL HCHAR(18,16,113,3):: CALL HCHAR(18,22,113,3)
730 CALL HCHAR(18,28,113,5):: CALL HCHAR(20,1,113,2):: CALL HCHAR(22,1,113,3)::
CALL VCHAR(18,2,124,2):: CALL VCHAR(20,3,124,2)
740 CALL VCHAR(8,3,124,10):: CALL VCHAR(11,10,124,2):: CALL VCHAR(11,11,124,4)::
 CALL VCHAR(8,12,124,3):: CALL VCHAR(15,12,124,3)
750 CALL VCHAR(6,15,124,2):: CALL VCHAR(16,16,124,2):: CALL VCHAR(16,18,124,2)::
 CALL VCHAR(6,19,124,2):: CALL VCHAR(11,20,124,3)
760 CALL VCHAR(5,23,124,3):: CALL VCHAR(10,23,124,4):: CALL VCHAR(8,24,124,2)::
CALL VCHAR(15,24,124,3):: CALL VCHAR(15,28,124,3)
770 CALL VCHAR(5,31,124,5):: CALL VCHAR(10,32,124,8):: CALL HCHAR(11,16,113,4)
780 CALL SPRITE(#2,140,14,121,60,#3,140,14,97,188,#4,136,4,81,80,#5,136,4,65,198
)
790 CALL SPRITE(#6,132,7,41,70,#7,132,7,17,208)
800 CALL MOTION(#2,0,Q1,#3,0,Q6,#4,0,Q2,#5,0,Q2):: CALL MOTION(#6,0,Q1,#7,0,Q4):
: RETURN
810 ON ERROR 1480
820 ! AUFSTELLUNG DER FRESSERTEILE
830 FOR I=1 TO 3 :: IF I=1 THEN 840 ELSE IF I=2 THEN 850 ELSE 860
840 P1=A(1):: P2=A(2):: P3=A(3):: P4=AA :: GOTO 870
850 P1=B(1):: P2=B(2):: P3=B(3):: P4=BB :: GOTO 870
```




```
860 P1=C(1):: P2=C(2):: P3=C(3):: P4=CC :: GOTO 870
870 DISPLAY AT(P1,P4-3)SIZE(3):"abc" :: DISPLAY AT(P2,P4-3)SIZE(3):"hij" :: DISP
LAY AT(P3,P4-3)SIZE(3):"def" :: NEXT I
880 ! STEUERRUNG
890 GOSUB 1050 :: CALL POSITION(#1,Y,X):: CALL JOYST(1,XX,YY):: CALL KEY(1,K,S)
900 IF YY<>0 AND XX<>0 THEN X2=0
910 IF K<>-1 THEN X1=XX :: IF XX=4 THEN 1010 ELSE IF XX=-4 THEN 1030 ELSE 1000
920 IF YY<>0 OR XX<>0 THEN 950
930 IF X1=-4 THEN CALL PATTERN(#1,36)ELSE CALL PATTERN(#1,40)
940 GOTO 890
950 GOSUB 1050 :: IF XX=-4 THEN X=X-8 :: X1=XX :: CALL PATTERN(#1,116)ELSE IF XX
=4 THEN X=X+8 :: X1=XX :: CALL PATTERN(#1,128)ELSE 970
960 GOTO 1060
970 IF YY=4 THEN 1150 ELSE IF YY=-4 THEN 1160
980 GOTO 1060
990 ! SPRUENGE
1000 FOR I=14 TO -14 STEP -2 :: CALL MOTION(#1,-I,0):: FOR II=1 TO 8 :: NEXT II
:: NEXT I :: CALL MOTION(#1,0,0):: GOTO 1060
1010 CALL PATTERN(#1,128):: FOR I=12 TO -12 STEP -2 :: CALL MOTION(#1,-I,5):: GO
SUB 1050 :: FOR II=1 TO 8 :: NEXT II :: NEXT I :: X=X+32
1020 CALL MOTION(#1,0,0):: GOTO 1060
1030 CALL PATTERN(#1,116):: FOR I=12 TO -12 STEP -2 :: CALL MOTION(#1,-I,-5):: G
OSUB 1050 :: FOR II=1 TO 8 :: NEXT II :: NEXT I :: X=X-32
1040 CALL MOTION(#1,0,0):: GOTO 1060
1050 CALL COINC(ALL,CF):: IF CF=-1 THEN 1480 :: RETURN
1060 AE=INT((X+10)/8):: IF AE>32 THEN X=X-8 ELSE IF AE<2 THEN X=X+8
1070 GOSUB 1050 :: CALL LOCATE(#1,Y,X):: Y=INT((Y+15)/8):: X=INT((X+10)/8):: CAL
L GCHAR(Y,X,G)
1080 IF G=98 OR G=101 OR G=105 THEN 1230 ELSE IF G=122 OR G=123 THEN 1460
1090 CALL GCHAR(Y+1,X,G):: IF G=113 OR G=124 THEN 890 ELSE 1480
1100 IF L=0 THEN L=1 ELSE L=0
1110 ON L+1 GOTO 1120,1130
1120 CALL HCHAR(12,22-BI,120):: CALL HCHAR(12,23-BI,121):: CALL HCHAR(13,22-BI,1
22):: CALL HCHAR(13,23-BI,123):: RETURN
1130 CALL HCHAR(12,22-BI,32,2):: CALL HCHAR(13,22-BI,32,2):: RETURN
1140 CALL HCHAR(4,16,32,2):: CALL HCHAR(5,16,32,2):: RETURN
1150 Y=Y-8 :: YB=INT((Y+15)/8):: XB=INT((X+10)/8):: CALL GCHAR(YB,XB,G):: IF G<>
124 THEN Y=Y+8 :: GOTO 890 ELSE 1170
1160 Y=Y+8 :: YY=INT((Y+15)/8):: XX=INT((X+10)/8):: CALL GCHAR(YY,XX,G):: IF G<>
124 THEN Y=Y-8 :: GOTO 890
1170 GOSUB 1050 :: CALL PATTERN(#1,60):: CALL LOCATE(#1,Y,X):: CALL POSITION(#1,
Y,X):: YB=INT((Y+15)/8):: XB=INT((X+10)/8)
1180 CALL GCHAR(YB,XB,G):: IF G<>124 THEN 890 ELSE CALL GCHAR(YB+1,XB,G):: IF G<
>124 THEN 890
1190 ! STEUERRUNG 2
1200 CALL POSITION(#1,Y,X):: CALL JOYST(1,XX,YY):: IF YY=4 THEN Y=Y-8 ELSE IF YY
=-4 THEN Y=Y+8
1210 GOTO 1170
1220 ! SETZEN DER FRESSERTEILE
1230 IF X=AA THEN 1240 ELSE IF X=BB THEN 1250 ELSE 1260
1240 XY=A(1):: YX=A(2):: AX=AA :: GOTO 1270
1250 XY=B(1):: YX=B(2):: AX=BB :: GOTO 1270
1260 XY=C(1):: YX=C(2):: AX=CC
1270 CALL SOUND(100,345,0):: IF G=98 THEN 1280 ELSE IF G=105 THEN 1380 ELSE 1420
1280 GOSUB 1100 :: IF Y-10=XY THEN 1290 ELSE IF Y-5=XY THEN 1320 ELSE 1340
1290 SC=SC+50 :: CALL HCHAR(Y,X-1,32,3):: DISPLAY AT(20,AX-3)SIZE(3):"abc"
1300 CALL GCHAR(20,7,G):: CALL GCHAR(20,17,HI):: CALL GCHAR(20,27,GV):: IF G=32
OR HI=32 OR GV=32 THEN 1430
1310 GOSUB 1450 :: CALL POSITION(#21,Y,X):: SC=SC+(X*BI*3*PH):: GOTO 220
1320 SC=SC+50 :: CALL HCHAR(Y,X-1,32,3):: DISPLAY AT(Y+5,X-3)SIZE(3):"abc"
1330 DISPLAY AT(21,AX-3)SIZE(3):"hij" :: GOTO 1430
1340 CALL GCHAR(Y+5,X,G):: CALL GCHAR(Y+10,X,GH):: IF G=32 AND GH=32 THEN 1350 E
LSE 1360
```

```
1350 SC=SC+50 :: CALL HCHAR(Y,X-1,32,3):: DISPLAY AT(Y+5,X-3)SIZE(3):"abc" :: GO
TO 1430
1360 SC=SC+100 :: CALL HCHAR(Y,X-1,32,3):: DISPLAY AT(Y+5,X-3)SIZE(3):"abc" :: D
ISPLAY AT(Y+10,X-3)SIZE(3):"hij"
1370 DISPLAY AT(22,AX-3)SIZE(3):"def" :: GOTO 1430
1380 IF Y-5=YX THEN 1390 ELSE 1400
1390 SC=SC+50 :: CALL HCHAR(Y,X-1,32,3):: DISPLAY AT(21,AX-3)SIZE(3):"hij" :: GO
TO 1430
1400 SC=SC+75 :: CALL HCHAR(Y,X-1,32,3):: CALL GCHAR(Y+5,X,G):: DISPLAY AT(Y+5,X
-3)SIZE(3):"hij" :: IF G=101 THEN 1410 ELSE 1430
1410 SC=SC+50 :: DISPLAY AT(22,AX-3)SIZE(3):"def" :: GOTO 1430
1420 SC=SC+50 :: CALL HCHAR(Y,X-1,32,3):: DISPLAY AT(22,AX-3)SIZE(3):"def"
1430 GOTO 890
1440 ! PUNKTANZEIGE
1450 SC$=STR$(SC):: FOR I=1 TO LEN(SC$):: CALL HCHAR(24,25+I,ASC(SEG$(SC$,I,1)))
:: NEXT I :: RETURN
1460 CALL SPRITE(#21,108,13,1,256,0,-1):: GOTO 890
1470 ! AUSWERTUNG,SPIELENDE?
1480 FOR I=1 TO 10 :: CALL SOUND(100,I*110,0):: NEXT I :: GOSUB 1450
1490 M=M-1 :: IF M=-1 THEN 1510 ELSE CALL HCHAR(2,2,32,7):: CALL HCHAR(2,2,95,M)
1500 CALL MOTION(#1,0,0):: CALL SPRITE(#20,128,16,1,40,#21,108,13,1,256,0,-1)::
CALL SPRITE(#1,128,16,153,6):: GOTO 890
1510 DISPLAY AT(12,7):"G A M E - O V E R"
1520 FOR I=5 TO 1 STEP -1 :: CALL SOUND(250,I*110,0,-8,6):: CALL SOUND(250,330,0
,110,7,234,10):: NEXT I
1530 CALL KEY(0,K,S):: IF S=0 THEN 1530 ELSE 200
1540 SUB MUSIK
1550 DATA 131,131,165,165,147,147,131,131,165,165,196,196,175,175,165,131,196,19
6,175,220,165,165,175,175,147,147
1560 RESTORE 1550 :: DIM Z(26):: FOR I=1 TO 26 :: READ Z(I):: NEXT I
1570 FOR I=1 TO 18 :: CALL SOUND(220,Z(I),0):: NEXT I :: CALL SOUND(1000,220,0):
: FOR I=19 TO 20 :: CALL SOUND(220,Z(I),0):: NEXT I
1580 CALL SOUND(1000,196,0):: FOR I=21 TO 26 :: CALL SOUND(220,Z(I),0):: NEXT I
:: CALL SOUND(1000,131,0)
1590 SUBEND
```

Cave-Man

Das Spiel ist auf dem TI 99/41 in TI BASIC geschrieben und wird mittels Joystick 1 gesteuert. Das Programm wurde so gestaltet, daß es auch in Extended BASIC ohne Änderungen läuft.

In Extended BASIC läuft das Programm jedoch fast doppelt so schnell als in TI BASIC. X-BASIC-Besitzer sollten das Programm auch eintippen. Es lohnt sich! Cave man zeichnet sich besonders durch die hervorragende High Resolution Grafik aus.

Schwer hatten es die Menschen in der Steinzeit, denn sie mußten sich ihr Essen bitter erkämpfen. Leo Neandertal ist nun einer dieser Menschen, der seine Familie ernähren muß.

Leo hat die Aufgabe, dem Saurier Rex die Eier zu stehlen und diese Eier in den unterirdischen Höhlen zu lagern. Freilich hat der Saurier Rex etwas gegen Leos Unterfangen und versucht mit allen Mitteln, Leo Neandertal am Diebstahl der Eier zu hindern.

Doch nun zum eigentlichen Spiel:

Tippen Sie das Programm ab, oder laden Sie dieses von Cassette oder Diskette ein.

Starten Sie das Programm jetzt mit <RUN>, und das Titelbild erscheint. Sie haben jetzt die Möglichkeit, zwischen 8 Schwierigkeitsstufen zu wählen, wobei 1 die kleinste Schwierigkeit und 8 die größte Schwierigkeit ist. Die Einstellung der Schwierigkeit erfolgt mittels Joystick und wird zu Programmbeginn vom Rechner selbst erklärt.

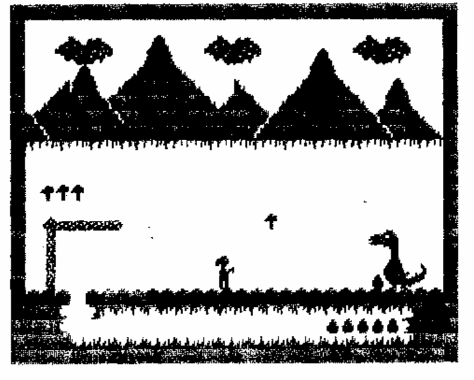
Nachdem das Spielbild erschienen ist, ertönt die ‚Cave-man'-Fanfare. Das ist das Zeichen, um die Schwierigkeit mittels Joystick zu wählen und danach den Aktionsknopf zu drücken, um das Spiel zu beginnen.

Leo Neandertal steht links im Bild unter seiner Hütte. Rechts im Bild ist der Saurier Rex zu sehen, welcher immer wieder den Kopf neigt und Feuer speiht. Steuern Sie Leo mittels des Joysticks nach rechts, bis Sie den Saurier Rex erreicht haben. Leo wird sich jetzt automatisch nach links umdrehen, und das Ei des Sauriers erscheint vor Leo. Bewegen Sie den Joystick

jetzt nach links, um Leo zusammen mit dem Ei in die Hütte zurückzuführen, damit Leo das Ei in die Höhlen werfen kann.
Sind sie jedoch im Bereich des Sauriers vorsichtig, da dieser Feuer speiht. Wenn Leo Neandertal von diesem Feuer erwischt wird, so verliert er sein Leben, und das Spiel ist zu Ende.
Leo hat jedoch eine Waffe gegen den Saurier Rex: seinen selbstgezimmerten Steinzeithammer. Leo kann diesen Hammer mitnehmen und diesen dem Saurier an den Kopf werfen, wodurch Rex für kurze Zeit betäubt wird. Dann kann Leo ungehindert das Ei stehlen. Zu Spielbeginn stehen Leo 3 Hämmer zur Verfügung.
Wenn Leo die Absicht hat, sich mit einem Hammer auf den Weg zu Rex zu machen, so muß er in seiner Hütte stehen und der Aktionsknopf muß gedrückt werden. Leo wendet sich jetzt dem Saurier zu, und der Hammer erscheint vor Leo. Sobald Leo den Hammer jedoch genommen hat, kann er nur noch Richtung Saurier gehen, aber nicht mehr zur Hütte zurück.
Will Leo jetzt den Hammer werfen, so ist wiederum der Aktionsknopf zu drücken. Es empfiehlt sich jedoch, schon während des Fluges des Hammers weiter in Richtung Saurier zu gehen, da die Ohnmacht nur kurz anhält.

Trifft der Hammer den Saurier am Kopf, so sieht der gute Rex jetzt Sternchen. Wenn der Hammer jedoch nicht trifft oder Leo den Saurier erreicht hat, ohne den Hammer zu werfen, so wird ein Hammer abgezogen. Wenn kein Hammer mehr zur Verfügung steht, so ist das Spiel beendet.

Sobald Leo sein Leben verloren hat, ertönt ein fürchterlicher Todesschrei unseres armen Leos. Danach ertönt die ‚Cave-man'-Fanfare, und Sie können durch Druck auf den Aktionsknopf das Spiel neu beginnen.
Wenn Leo das Ei gestohlen hat, so kann er nur noch Richtung Hütte gehen und nicht mehr in die Richtung des Sauriers.
Schwierigkeitssteigerung:
Die anfänglich gewählte Schwierigkeitsstufe steigert sich nach jeweils 11 gesammelten Eiern.
Wenn es Ihnen gelingt, die gesamte Höhle mit Eiern zu füllen (22 Eier), so bekommen Sie einen Bonushammer (maximal können 6 Hämmer angezeigt werden).
Punktwertung:
Sie erhalten für jedes gesammelte Ei 10mal Schwierigkeitsstufe Punkte. Gelingt es Ihnen, die gesamte Höhle mit Eiern zu füllen, so werden Ihnen neben dem Bonushammer zusätzlich 1000mal der augenblicklichen Schwierigkeit Punkte gutgeschrieben.
Nachdem das Spiel zu Ende ist, können Sie durch Druck auf den Aktionsknopf ein neues Spiel beginnen.
Tips zum Eintippen:
Extended-BASIC-Besitzer sollten zur Eingabeerleichterung das Programm am TI BASIC-Modus eintippen, da die aufeinanderfolgenden Doppelpunkte in den Print-Anweisungen sonst als Multi-Statement-Seperatoren vom Rechner verarbeitet werden könnten.
Beim Spielablauf sollte die <ALPHA LOCK>-Taste nicht eingerastet werden, da sonst die Joystickabfrage nicht richtig funktioniert.
Wir wünschen Ihnen ein ERRORfreies Eintippen und viel Spaß beim Spiel!



```
100 REM **************************
110 REM *    C A V E   M A N    *
120 REM *-----------------------*
130 REM * TI 99/4 A + JOYSICK 1  *
140 REM * Programmsprache :      *
150 REM * T I   B A S I C        *
160 REM *-----------------------*
170 REM * (c) 1984 by Tronicsoft *
180 REM *              V.Becker  *
190 REM **************************
200 REM
210 REM
220 CALL CLEAR
230 CALL SCREEN(2)
240 FOR I=1 TO 14
250 CALL COLOR(I,2,1)
260 NEXT I
270 GOSUB 3330
280 GOSUB 3470
290 RU=1
300 P=0
310 AH=3
320 SE=0
330 HF=0
340 AE=0
350 PRINT : :"  Prtv  ^b  Prtv       Prtv    qsuw  ac  qsuw   ^b qsuw "
360 PRINT "   ^b  exxd          ac            eac exxxxd  d   exxd  ^b   exxxhjxxxxxd
exd exxxxd ac"
370 PRINT "hjxxxikxxxxxxxxlnxxxxxxhjxd ikxxxxxxxxxxxxxmoxxxxxxikxxd"
380 PRINT "f9f9f9ff9f99f99f9f9f9ff9f9f9"
390 FOR I=1 TO 4
400 PRINT
410 NEXT I
420 PRINT " XZZZZ[":" Y":" Y":" Y":" Y":"\]\ \\\\\\\\\\\\\\\\\\\\\\\\"
430 PRINT "VVU TRSRRSSSRSSRRRRSRSRRSRVVVVU                    TVVVVVPQQPPPQPQQ
PQPQQPPPQPQQPVV":
440 DATA 2,16,2,2,13,13,11,10,5,5,15,5,9,11
```

```
450 RESTORE 440
460 FOR I=1 TO 14
470 READ C
480 CALL COLOR(I,C,2)
490 NEXT I
500 CALL HCHAR(13,4,94,AH)
510 GOSUB 2470
520 GOSUB 2550
530 S=8
540 GOSUB 750
550 GOSUB 2800
560 GOSUB 3200
570 CALL VCHAR(19,26,40)
580 STV=1
590 FOR LI=1 TO 9-RU
600 CALL JOYST(1,X,Y)
610 CALL KEY(1,T,U)
620 IF (T=18)THEN 1690
630 IF HF=2 THEN 2010
640 X=X/4+2
650 ON X GOTO 820,660,980
660 IF (S>23)*(FI=1)THEN 1900
670 NEXT LI
680 IF SB>0 THEN 2310
690 STV=STV+1
700 IF STV>4 THEN 710 ELSE 720
710 STV=1
720 ON STV GOSUB 2550,2610,2670,2750
730 IF (S>23)*(FI=1)THEN 1900
740 GOTO 590
750 CALL HCHAR(24,1,78,64)
760 CALL VCHAR(1,31,78,96)
770 CALL COLOR(3,2,13)
780 CALL COLOR(4,2,13)
790 GOSUB 2990
800 GOSUB 3050
810 RETURN
820 IF HF<>0 THEN 660
830 IF (S=7)*(SE=1)THEN 1300
840 IF S=7 THEN 660
850 IF SE=1 THEN 910
860 GOSUB 2940
870 S=S-1
880 GOSUB 2870
890 IF SE=1 THEN 940
900 GOTO 660
910 CALL VCHAR(19,PE,32)
920 CALL SOUND(-10,261,0)
930 GOTO 860
940 PE=S-1
950 CALL VCHAR(19,PE,40)
960 CALL SOUND(-10,440,0)
970 GOTO 660
980 IF SE=1 THEN 660
990 IF S=25 THEN 1060
1000 IF HF=1 THEN 1150
1010 GOSUB 2940
1020 S=S+1
1030 IF HF=1 THEN 1180
1040 GOSUB 2800
1050 GOTO 660
1060 SE=1
1070 GOSUB 2940
1080 GOSUB 2870
1090 CALL VCHAR(19,26,32)
1100 CALL SOUND(-10,261,0)
1110 PE=S-1
1120 CALL HCHAR(19,PE,40)
1130 CALL SOUND(-10,440,0)
1140 GOTO 660
1150 CALL HCHAR(18,HP,32)
1160 CALL SOUND(-10,698,0)
1170 GOTO 1010
1180 HP=S+1
1190 IF HP>24 THEN 1200 ELSE 1260
1200 CALL VCHAR(13,3+AH,32)
1210 CALL SOUND(-10,110,5,112,5,114,5,-7,0)
1220 AH=AH-1
1230 IF AH=0 THEN 1900
1240 HF=0
1250 GOTO 1040
1260 HP=S+1
1270 CALL VCHAR(18,HP,94)
1280 CALL SOUND(-10,880,0)
1290 GOTO 1040
1300 FOR Z=19 TO 21
1310 CALL VCHAR(Z,6,32)
1320 CALL SOUND(-5,261,0)
1330 CALL VCHAR(Z+1,6,40)
1340 CALL SOUND(-5,440,0)
1350 CALL SOUND(-5,1046,0)
1360 NEXT Z
1370 FOR Z=6 TO 26-AE
1380 CALL VCHAR(22,Z,32)
1390 CALL SOUND(-5,261,0)
1400 CALL VCHAR(22,Z+1,40)
1410 CALL SOUND(-5,440,0)
1420 CALL SOUND(-5,523,0)
1430 NEXT Z
1440 SE=0
1450 AE=AE+1
1460 P=P+10*RU
1470 GOSUB 2990
1480 GOSUB 1520
1490 CALL VCHAR(19,26,40)
1500 CALL SOUND(-10,440,0)
1510 GOTO 660
1520 IF AE=11 THEN 1640
1530 IF AE=22 THEN 1550
1540 RETURN
1550 AH=AH+1
1560 IF AH>6 THEN 1570 ELSE 1580
1570 AH=6
1580 CALL HCHAR(13,4,94,AH)
1590 GOSUB 3200
1600 P=P+1000*RU
1610 GOSUB 2990
1620 CALL HCHAR(22,6,32,22)
1630 AE=0
1640 RU=RU+1
1650 IF RU>8 THEN 1660 ELSE 1670
1660 RU=8
1670 GOSUB 3290
1680 RETURN
1690 IF (SE=1)+(HF=2)THEN 640
1700 IF HF=1 THEN 1800
1710 IF (HF=0)*(S<9)THEN 1730
1720 GOTO 640
1730 GOSUB 2940
1740 GOSUB 2800
1750 HF=1
1760 HP=S+1
1770 CALL VCHAR(18,HP,94)
1780 CALL SOUND(-10,880,0)
1790 GOTO 640
1800 IF S<8 THEN 640
1810 FW=(12/(9-RU))/8
1820 HF=2
1830 CALL VCHAR(18,HP,32)
1840 CALL SOUND(-10,698,0)
```

```
1850 SH=1
1860 FH=18-SH
1870 CALL VCHAR(FH,HP,94)
1880 CALL SOUND(-10,880,0)
1890 GOTO 640
1900 FOR I=1100 TO 500 STEP -10
1910 CALL SOUND(-100,I,0,I-2,0,I+2,0)
1920 NEXT I
1930 GOSUB 3200
1940 CALL KEY(1,T,U)
1950 IF U=0 THEN 1940
1960 CALL CLEAR
1970 FOR I=1 TO 14
1980 CALL COLOR(I,2,1)
1990 NEXT I
2000 GOTO 290
2010 CALL VCHAR(FH,HP,32)
2020 CALL SOUND(-10,698,0)
2030 IF FH=11 THEN 2040 ELSE 2050
2040 SH=-SH
2050 FH=FH-SH
2060 HP=HP+FW
2070 CALL VCHAR(FH,HP,94)
2080 CALL SOUND(-10,880,0)
2090 IF (HP>26)*(FH>15)THEN 2220
2100 IF HP>27 THEN 2130
2110 IF FH>18 THEN 2130
2120 GOTO 640
2130 CALL VCHAR(FH,HP,32)
2140 CALL SOUND(-10,698,0)
2150 CALL VCHAR(13,3+AH,32)
2160 CALL SOUND(-10,110,5,112,5,114,5,-7,0)
2170 CALL VCHAR(19,26,40)
2180 AH=AH-1
2190 IF AH=0 THEN 1900
2200 HF=0
2210 GOTO 640
2220 HF=0
2230 FI=0
2240 SB=6
2250 CALL VCHAR(FH,HP,32)
2260 CALL SOUND(-10,698,0)
2270 GOSUB 2750
2280 GOSUB 2610
2290 STV=2
2300 GOTO 640
2310 CALL HCHAR(15,26,32,2)
2320 STS=STS+1
2330 IF STS>2 THEN 2340 ELSE 2350
2340 STS=1
2350 ON STS GOTO 2400,2430
2360 SB=SB-1
2370 IF SB=0 THEN 2380 ELSE 740
2380 CALL HCHAR(15,26,32,2)
2390 GOTO 740
2400 CALL HCHAR(15,26,140,2)
2410 CALL SOUND(-10,1396,0)
2420 GOTO 2360
2430 CALL VCHAR(15,26,141)
2440 CALL VCHAR(15,27,143)
2450 CALL SOUND(-10,1046,0)
2460 GOTO 2360
2470 REM Saurier Korpus
2480 CALL VCHAR(18,27,72)
2490 CALL VCHAR(19,27,73)
2500 CALL VCHAR(18,28,74)
2510 CALL VCHAR(19,28,75)
2520 CALL VCHAR(18,29,76)
2530 CALL VCHAR(19,29,77)
2540 RETURN
2550 REM Saurier Kopf gerade
2560 CALL VCHAR(16,26,68)
2570 CALL VCHAR(17,26,69)
2580 CALL VCHAR(16,27,70)
2590 CALL VCHAR(17,27,71)
2600 RETURN
2610 REM Saurier Kopf geneigt
2620 CALL VCHAR(16,26,64)
2630 CALL VCHAR(17,26,65)
2640 CALL VCHAR(16,27,66)
2650 CALL VCHAR(17,27,67)
2660 RETURN
2670 REM Feuer des Saurier
2680 FI=1
2690 CALL VCHAR(18,24,132)
2700 CALL VCHAR(19,24,133)
2710 CALL VCHAR(18,25,134)
2720 CALL VCHAR(19,25,135)
2730 CALL SOUND(-10,245,0,246,0,247,0,-5,0)
2740 RETURN
2750 REM Feuer loeschen
2760 FI=0
2770 CALL VCHAR(18,24,32,2)
2780 CALL VCHAR(18,25,32,2)
2790 RETURN
2800 REM Cave Man nach rechts
2810 CALL VCHAR(18,S,128)
2820 CALL VCHAR(19,S,129)
2830 CALL VCHAR(17,S,137)
2840 CALL VCHAR(18,S-1,138)
2850 CALL SOUND(-10,523,0)
2860 RETURN
2870 REM Cave Man nach links
2880 CALL VCHAR(18,S,130)
2890 CALL VCHAR(19,S,131)
2900 CALL VCHAR(17,S,139)
2910 CALL VCHAR(18,S+1,136)
2920 CALL SOUND(-10,523,0)
2930 RETURN
2940 REM Cave Man loeschen
2950 CALL VCHAR(17,S,32,3)
2960 CALL HCHAR(18,S-1,32,3)
2970 CALL SOUND(-10,349,0)
2980 RETURN
2990 P$=STR$(P)
3000 FOR LG=1 TO LEN(P$)
3010 PR=ASC(SEG$(P$,LG,1))
3020 CALL HCHAR(1,30-LEN(P$)+LG,PR)
3030 NEXT LG
3040 RETURN
3050 CALL VCHAR(18,7,ASC(STR$(RU)))
3060 GOSUB 3200
3070 CALL JOYST(1,X,Y)
3080 IF (X=0)*(Y=0)THEN 3130
3090 RU=RU+1
3100 IF RU>8 THEN 3110 ELSE 3120
3110 RU=1
3120 CALL VCHAR(18,7,ASC(STR$(RU)))
3130 FOR VERZ=1 TO 10
3140 CALL KEY(1,T,U)
3150 IF T=18 THEN 3180
3160 NEXT VERZ
3170 GOTO 3070
3180 CALL VCHAR(18,7,32)
3190 RETURN
3200 CALL SOUND(200,523,0,659,0)
3210 CALL SOUND(100,391,0)
3220 CALL SOUND(100,440,0)
3230 CALL SOUND(200,523,0)
3240 CALL SOUND(200,587,0)
```

```
3250 CALL SOUND(200,523,0)
3260 CALL SOUND(200,329,0,391,0)
3270 CALL SOUND(200,391,0,523,0)
3280 RETURN
3290 CALL SOUND(200,391,0,523,0,659,0)
3300 CALL SOUND(200,440,0,523,0,698,0)
3310 CALL SOUND(200,391,0,523,0,659,0)
3320 RETURN
3330 PRINT "      C A V E   M A N              -----------------": : :
3340 PRINT "Schwierigkeitswahl :": :"Druecke roten Aktionsknopf. Warte bis Fanfa
re ertoent."
3350 PRINT "Bewege Joystick 1 so lange, bis die gewuenschte"
3360 PRINT "Schwierigkeit im gruenen    Feld unter der Huette       erscheint."
3370 PRINT "Lasse dann Joystick los und druecke erneut Aktionsknopf.": : :
3380 PRINT "1 - kleinste Schwierigkeit  8 - groesste Schwierigkeit": : :"    @ 1
984 by Tronicsoft"
3390 CALL CHAR(64,"3C4299A1A199423C")
3400 CALL SCREEN(4)
3410 GOSUB 3200
3420 CALL KEY(1,T,U)
3430 IF U=0 THEN 3420
3440 CALL SCREEN(2)
3450 CALL CLEAR
3460 RETURN
3470 REM Grafik umbelegen
3480 DATA ,0010101010100001,00282828,0028287C287C2828,0038545038145438,0060640810
204C0C,0020505020544834
3490 DATA 0008081,000C1E1E3F3F3F1E,0020100808081020,000028107C1028,000010107C101,
000000000030102,000000007C
3500 DATA 000000000000303,0000040810204,0018242424242418,0008180808080808,0018240408
103C,0018240804241 8,0028282838080808
3510 DATA 003C2038042418,0018203824241 8,003C040810101,0018241824241 8,001824241C0
418,0000303000303,000030300030102
3520 DATA 0008102040201008,0000007C007C,0020100804081020,0038440408100010,00000001
03060C0F,1F1F3F3E7C78E04,3878FCFEFF7FFFFF
3530 DATA FFFF9F0F0F0F0F0F,0000001F7FFF3FFF,7F,0078FCFE9FFFFFFF,FFFF1F0F0F0F0F0F
,0F1F1F3F7FFFFFFF,FF7F7F3F1F0F060E
3540 DATA 008080C0C0C0E0E8,FCF8F1FFFFFF367,000000080C06060E,5E7CFCF8F08,FFFFFFFF
FFFFFFFF,,00000282A3E7FFFF
3550 DATA 000001092B6FFFFF,FFFFE7C5414,FFFFF6D4908,0F3F0703030F071F,F8E0F0C0C0E0
FCF,FFFFFFFFFFFFFFFF,0044444454545428
3560 DATA 00040F1B3E6F3D17,171F1B1F161F0B1D,000000BFEEBB7FEF,000000C060F0A0C,187
EFFFFFFFFFFFF,1F7FFFFFFFFFFFFF,183E7EFE18181818
3570 DATA 000000000000007C,010307070F0F1F1F,3F7F7F3F7F7FFFFF,80C0E0E0E0F0F8F8,F8
FCFCFEFCFEFFFF,80E0E0F0F4FCFEFF,0107070F2F3F7FFF
3580 DATA FFB9B1111,D3921202,0387EFF7FBFBF9FE,FFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFF,
7F3FDFDFCFF7F7FF,FFFFFFFFFFFFFFFF
3590 DATA FEFCFBFBF3EFEFFF,C0E1F7EFDFDF9F7F,FFFFFFFFFFFFFFFF,0001010101071F3DFF,FF
FF1F070703,00C0E0F8FCDFFFFF,DDFFFFEFFF3F1F0C
3600 DATA 0000337F7FFFFFFF,FBFFFFEFFFFCF88,00008080E0F8BCFE,F6FE7EECE0C08,FFFFFF
FFFFFFFFFF,000000442810101,0000007C0810207C
3610 DATA 0018202040202018,0010101000101010,0030080804080830,0000205408,,E8FCF8706
177FDF8,F8F8F8F8D8D8D8FC
3620 DATA 173F1F0E86EEBF1F,1F1F1F1F1B1B1B3F,,0103070F1F3F7FFF,0103060C1C3878F,F0
E0E0C0C0808,80C0C0C
3630 DATA 0000000000070F8,01030303,000000000000E1F,0000004428102844,0000080402
030408,,000020408080402
3640 RESTORE 3480
3650 FOR I=32 TO 143
3660 READ C$
3670 CALL CHAR(I,C$)
3680 NEXT I
3690 RETURN
```

Expedition

```
   1 REM >>EXPEDITION<<
        (C) BY RALF JAHNKE 8/84
   4 CLS
   5 LET P=150
   6 LET T=20
   7 LET S=0
   8 GOSUB 600
   9 GOSUB 450
  10 LET X=10
  20 LET Y=10
  30 PRINT AT X,Y;"■"
  40 PRINT AT X,Y;"▒"
  50 LET X=X+(INKEY$="6")-(INKEY
$="7")
  60 LET Y=Y+(INKEY$="8")-(INKEY
$="5")
  62 IF X>=21 OR X<0 OR Y>=32 OR
 Y<0 THEN GOSUB 1000
  65 PRINT AT 10,10;"C"
  70 IF X=5 AND Y=12 THEN GOTO 2
500
  80 IF X=10 AND Y=3 THEN GOTO 3
500
  90 IF X=15 AND Y=6 THEN GOTO 4
000
 100 IF X=20 AND Y=9 THEN GOTO 4
500
 110 IF X=8 AND Y=15 THEN GOTO 5
000
 120 IF X=16 AND Y=18 THEN GOTO
5500
 130 IF X=2 AND Y=21 THEN GOTO 6
000
 140 IF X=17 AND Y=24 THEN GOTO
6500
 150 IF X=9 AND Y=1 THEN GOTO 70
00
 160 IF X=12 AND Y=5 THEN GOTO 7
500
 170 IF X=3 AND Y=4 THEN GOTO 80
00
 180 IF X=6 AND Y=7 THEN GOTO 85
00
 190 IF X=6 AND Y=27 THEN GOTO 9
000
 200 IF X=8 AND Y=19 THEN GOTO 3
000
 210 GOTO 30
 450 CLS
 497 IF P<=0 THEN GOTO 1600
 498 IF T<=0 THEN GOTO 2000
 499 IF S>=10 THEN GOTO 1500
 500 PRINT
"
```

Ein Abenteuerspiel mit guter Grafik.

Sie befinden sich auf der Südseeinsel „Camazulou" und leiten eine dortige Expedition.

In Ihrer Begleitung befinden sich Eingeborene, die als Lastenträger für Ihre Ausrüstung und Verpflegung ausgesucht wurden. Von Ihrem Freund Harry haben Sie eine Karte bekommen, auf der einige wichtige Stellen der Insel markiert sind. Harry hat aber nicht vermerkt, was Sie dort finden werden. Im Südwesten der Insel haben Sie Ihr Camp ›C‹ errichtet und starten von dort aus, mit Hilfe der Cursortasten, Ihre Expeditionen.

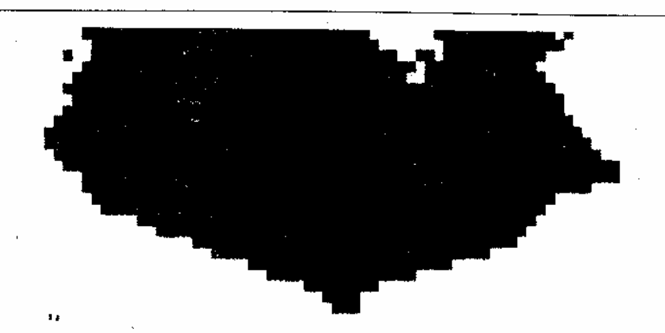


Ziel des Spieles ist es, möglichst viele der eingetragenen Schatzpunkte anzulaufen und die Funde einzusammeln. Passen Sie aber auf, daß keiner Ihrer Begleiter verlorengeht und Ihr Proviant bis zum Ziel ausreicht.

Korrektur
ZX-Spectrum
Ausgabe 10

Oil-Panic

340 FOR c = 6 TO 26 STEP 4 : BEEP .01,35 : PRINT AT 17, c; "ä"; AT 18, c; "ä"; AT 19, c; "ä"; AT 20, c; "ä": NEXT c



```
"
 505 PRINT AT 5,12;"█"
 510 PRINT AT 10,3;"█"
 515 PRINT AT 15,6;"█"
 520 PRINT AT 20,9;"."
 525 PRINT AT 8,15;"█"
 530 PRINT AT 16,18;"█"
 535 PRINT AT 2,21;"█"
 540 PRINT AT 17,24;"█"
 545 PRINT AT 9,1;"."
 550 PRINT AT 12,6;"█"
 555 PRINT AT 3,4;"█"
 560 PRINT AT 6,7;"█"
 565 PRINT AT 6,27;"█"
 570 PRINT AT 8,19;"."
 575 PRINT AT 21,0;"TRAEGER:";T
 580 PRINT AT 21,17;"PROVIANT:";
P
 585 PRINT AT 0,0;"SCHAETZE:";S
 590 RETURN
 600 CLS
 610 FOR N=1 TO 21
 620 PRINT "████████████████████
████████████"
 630 NEXT N
 640 PRINT AT 4,1;"    ";AT 5,1;"
";AT 6,1;"    ";AT 7,1;" ";AT 8,1
;"    "
 650 FOR N=0 TO 20
 660 NEXT N
 670 PRINT AT 4,5;" ██ ";AT 5,5;
"▌█▐";AT 6,5;"█ █";AT 7,5;"▌█
▐";AT 8,5;" ██ "
 680 FOR N=0 TO 20
 690 NEXT N
 700 PRINT AT 4,10;"    ";AT 5,10
;" █ ";AT 6,10;"    ";AT 7,10;" "
;AT 8,10;"  "
 710 FOR N=0 TO 20
 720 NEXT N
 730 PRINT AT 4,14;"    ";AT 5,14
;" ";AT 6,14;"   ";AT 7,14;" ";AT
8,14;" "
 740 FOR N=0 TO 20
 750 NEXT N
 760 PRINT AT 4,18;"    ";AT 5,18
;" ██ ";AT 6,18;" ██ ";AT 7,18;"
██  ";AT 8,18;"    "
 770 FOR N=0 TO 20
 780 NEXT N
 790 PRINT AT 4,23;" ";AT 5,23;"
";AT 6,23;" ";AT 7,23;" ";AT 8,
23;"  "
 800 FOR N=0 TO 20
 810 NEXT N
 820 PRINT AT 6,25;"    "
 830 FOR N=0 TO 20
 840 NEXT N
 850 PRINT AT 11,5;"    ";AT 12,
6;"▌▐";AT 13,6;"▌▐";AT 14,6;"▌▐"
;AT 15,6;"▌▐"
 860 FOR N=0 TO 20
 870 NEXT N
 880 PRINT AT 11,10;" ";AT 12,10
;" ";AT 13,10;" ";AT 14,10;" ";A
T 15,10;"  "
 890 FOR N=0 TO 20
 900 NEXT N
 910 PRINT AT 11,12;"    ";AT 12,
12;" █ ";AT 13,12;" █ ";AT 14,12
;" █ ";AT 15,12;"  "
 920 FOR N=0 TO 20
 930 NEXT N
 940 PRINT AT 11,16;" █ ";AT 12
,16;" █ ";AT 13,16;" █ ";AT 14
,16;" █ ";AT 15,16;" █ "
 950 FOR N=0 TO 20
 960 NEXT N
 970 PRINT AT 2,5;"FUER DEN ZX-8
1 (16 K)"
 980 PRINT AT 19,5;"(C) BY RALF J
AHNKE 8/84"
 990 FOR N=0 TO 300
 995 NEXT N
 999 GOTO 1100
1000 PRINT AT 10,2;"DA DIE KARTE
 ZUENDE IST,";AT 11,2;"MUESSEN S
IE UMKEHREN."
1010 FOR N=0 TO 100
1020 NEXT N
1030 GOTO 9
1100 CLS
1110 PRINT "
********************************
*  SIE BEFINDEN SICH AUF DER    *
*  SUEDSEEINSEL >CANAZULOU<,UND *
*  LEITEN EINE EXPEDITION.      *
"
1120 PRINT "
*  IN IHRER BEGLEITUNG BEFINDEN *
*  SICH EIN PAAR EINGEBOHRENE,  *
*  DIE IHNEN ALS TRAEGER FUER   *
*  IHRE AUSRUESTUNG DIENEN.     *
"
1130 PRINT "
*  VON IHREM FREUND HARRY HABEN *
*  SIE EINE KARTE MITBEKOMMEN,  *
*  AUF DER EINIGE STELLEN       *
*  MARKIERT SIND. JEDOCH KONNTE *
*  IHR FREUND IHNEN NICHT MEHR  *
*  MITTEILEN, WAS SIE AN DIESEN *
*  STELLEN FINDEN WERDEN.       *
"
1140 PRINT "
*  SIE HABEN IM SUEDWESTEN DER  *
*  INSEL IHR CAMP(C) ERRICHTET  *
*  UND STARTEN VON DORT AUS MIT *
*  HILFE DER KURSOR-TASTEN IHRE *
*  EXPEDITIONEN.                *
*  ZUM START EINE TASTE DRUECKEN*
********************************
"
1150 IF INKEY$="" THEN GOTO 1150
1160 RETURN
1500 CLS
1510 PRINT AT 3,10;"GRATULIERE";
AT 5,3;"SIE HABEN EINIGE SCHAETZ
E";AT 7,3;"ANGESAMMELT,UND KOENN
EN";AT 9,3;"STOLZ ZURUECK NACH H
AUS";AT 11,3;"FAHREN,UND VON IHR
EN";AT 13,3;"ABENTEUERN BERICHTE
N."
1520 PRINT AT 15,3;"SIE KOENNEN
ABER AUCH";AT 17,3;"EINE WEITERE
 EXPEDITION";AT 19,3;"MIT >E< ST
ARTEN, ODER";AT 21,3;"DIESES PRO
GRAMM MIT >S< SAVEN"
1525 IF INKEY$="" THEN GOTO 1525
1530 IF INKEY$="E" THEN RUN
1540 IF INKEY$="S" THEN GOTO 999
0
1550 STOP
1600 CLS
1610 PRINT AT 2,3;"ES IST ZUM HE
ULEN....";AT 4,3;"SIE WAREN SCHO
N SO WEIT";AT 6,3;"GEKOMMEN.";AT
 8,3;"ABER DER PROVIANT IST";AT
10,3;"VERBRAUCHT UND SIE MUESSEN
";AT 12,3;"EHLENDIG VERHUNGERN."
;AT 18,3;"FRIEDE IHRER ASCHE....
"
1620 GOTO 1620
2000 CLS
2010 PRINT AT 2,3;"ALLE IHRE TRA
EGER SIND";AT 4,3;"TOT......";A
T 8,3;"SIE MUESSEN OHNE IHRE";AT
 10,3;"AUSRUESTUNG NACH HAUS ZUR
```

```
UECK";AT 14,3;"WELCH EINE BLAMAG
E....";AT 16,3;"WENN SIE WOLLEN,
KOENNEN";AT 18,3;"SIE ABER MIT >
E< EINE";AT 20,3;"NEUE EXPEDITIO
N STARTEN."
2020 IF INKEY$="" THEN GOTO 2020
2030 IF INKEY$="E" THEN RUN
2040 STOP
2500 CLS
2510 PRINT "

"
2520 FOR N=1 TO 40
2525 PRINT AT 1,15;"O";AT 1,15;"
"
2527 NEXT N
2530 PRINT AT 6,0;"
  SIE SIND IN EINE HOEHLE
  GELANGT UND ENTDECKEN AN
  DER HOECHSTEN STELLE EINEN
  GLITZENDEN DIAMANTEN AN DER
  DECKE.                         "
2535 PRINT "
   WAS WOLLEN SIE TUN,UM AN
   DIESEN DIAMANTEN HERANZU-
   KOMMEN ? "
2537 PRINT "
          1... MIT IHREN TRAEGERN
               EINEN MENSCHENTURM
               BAUEN
          2... ZUM LAGER ZURUECK-
               GEHEN,UM EINE LEITER
               ZU HOHLEN
          3... DIE FEUCHTEN HOEHLEN-
               WAENDE HINAUFSTEIGEN
"
2540 IF INKEY$="" THEN GOTO 2540
2550 IF INKEY$="1" THEN GOTO 260
0
2560 IF INKEY$="2" THEN GOTO 270
0
2570 IF INKEY$="3" THEN GOTO 280
0
2600 CLS
2610 PRINT AT 3,3;"SIE HABEN DEN
 DIAMANTEN";AT 5,3;"ERREICHT UND
 KASSIEREN";AT 7,3;"EINEN SCHATZ
PUNKT.";AT 9,3;"JEDOCH HABEN DIE
 BEIDEN";AT 11,3;"UNTERSTEN TRAE
GER DIESE";AT 13,3;"ANSTRENGUNG
NICHT UEBERLEBT."
2620 LET S=S+1
2630 LET T=T-2
2640 FOR N=0 TO 200
2650 NEXT N
2660 GOTO 9
2700 CLS
2710 PRINT AT 3,3;"SIE HABEN EIN
E LEITER";AT 5,3;"GEFUNDEN,UND K
ONNTEN";AT 7,3;"DEN DIAMANTEN BE
RGEN.";AT 9,3;"ABER WAEHREND SIE
 WEG WAREN,";AT 11,3;"HABEN IHRE
 TRAEGER 1/4 IHRES";AT 13,3;"PROV
IANTES VERTILGT."
2720 LET S=S+1
2730 LET P=INT P-(P/4)
2740 FOR N=0 TO 200
2750 NEXT N
2760 GOTO 9
2800 CLS
2810 PRINT AT 3,3;"SIE HABEN DEN
 DIAMANTEN";AT 5,3;"GERADE IN DE
R HAND,ALS";AT 7,3;"SIE ABRUTSCH
TEN. IHRE";AT 9,3;"TRAEGER KONNT
EN SIE";AT 11,3;"GERADE NOCH AUF
FANGEN,";AT 13,3;"JEDOCH VERLORE
N SIE DEN";AT 15,3;"GESAMTEN INH
ALT IHRER";AT 17,3;"TASCHEN."
2815 PRINT AT 19,3;"ES BLEIBT IH
NEN NUR DER";AT 21,3;"DIAMANT IN
 IHRER HAND."
2820 LET S=1
2830 FOR N=0 TO 200
2840 NEXT N
2850 GOTO 9
3020 CLS
3025 FOR N=1 TO 30
3030 PRINT AT 3,0;"

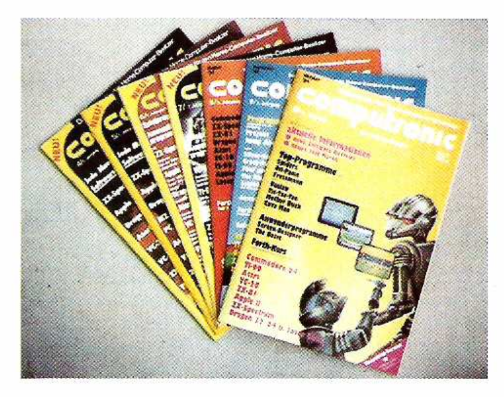
"
3040 PRINT AT 0,13;"";AT 1,1
4;"OO";AT 2,14;""
3050 PRINT AT 1,14;"OO";AT 2,14;
"";AT 7,0;"
"
3060 PRINT AT 10,3;"MITTEN IM SE
E IST EINE";AT 11,3;"MARKIERUNG
UND SIE SCHICKEN";AT 12,3;"EINIG
E TRAEGER ZUM TAUCHEN.";AT 14,3;
"ABER SIE FINDEN NICHTS,";AT 15,
3;"UND ZU GUTERLETZT WERDEN";AT
16,3;"NOCH 3 VON IHNEN VON EINEM
";AT 17,3;"SCHWARM KROKODILE GEF
RESSEN."
3070 NEXT N
3080 LET T=T-3
3090 GOTO 9
3500 CLS
3505 FOR N=1 TO 20
3510 PRINT AT 0,20;"";AT 1,1
9;"";AT 2,19;"";AT 3
,19;"";AT 4,20;""
3520 PRINT AT 0,20;"     ";AT 1,1
9;" ";AT 1,24;" ";AT 2,19;" ";AT
 2,24;" ";AT 3,19;" ";AT 3,24;"
";AT 4,20;"     "
3530 PRINT AT 8,0;"
"
3540 PRINT AT 4,12;" ";AT 5,11;
"";AT 6,12;" ";AT 7,11;""
3550 PRINT AT 11,3;"DIE SONNE HI
ER AM STRAND";AT 13,3;"SCHEINT E
RBARMUNGSLOS";AT 15,3;"UND IHR P
ROVIANT WIRD WENIGER";AT 17,3;"D
OCH DIE MUEHE LOHNT SICH.";AT 19
,3;"SIE FINDEN EINEN BEUTEL";AT
21,3;"VOLL GOLDMUENZEN."
3560 NEXT N
3570 LET S=S+1
3580 LET P=P-20
3590 GOTO 9
4000 CLS
4010 PRINT AT 3,3;"SIE GERATEN I
N EIN";AT 5,3;"SUMPFGEBIET UND V
ERLIEREN";AT 7,3;"DREI TRAEGER U
ND PROVIANT."
4020 LET T=T-3
4030 LET P=P-30
4040 FOR N=1 TO 150
4050 NEXT N
4060 GOTO 9
4500 CLS
4510 PRINT AT 3,0;"

"
4520 PRINT AT 15,3;"SIE ENTDECKE
N AUF DEM";AT 16,3;"MEERESBODEN
```

```
EIN SCHIFFS-";AT 17,3;"WRACK UND
 KOENNEN EINE";AT 18,3;"KISTE VO
LL PERLEN BERGEN.";AT 19,3;"JEDO
CH WURDEN ZWEI TRAEGER";AT 20,3;
"BEIM TAUCHEN VON HAIEN";AT 21,3
;"ZERRISSEN."
4530 LET S=S+1
4540 LET T=T-2
4550 FOR N=1 TO 200
4560 NEXT N
4570 GOTO 9
5000 CLS
5010 PRINT
     ████████████
           █ "
5020 PRINT AT 7,3;"MITTEN IM DSC
HUNGEL FINDEN";AT 8,3;"SIE EIN F
LUGZEUGWRACK AUS DEM";AT 9,3;"LE
TZTEN KRIEG.";AT 10,3;"SIE ENTDE
CKEN EINE KISTE";AT 11,3;"MIT WE
RTVOLLEN BRIEFMARKEN."
5030 LET S=S+1
5040 FOR N=1 TO 200
5045 NEXT N
5050 GOTO 9
5500 CLS
5510 PRINT AT 2,3;"EINE VERGESSE
NE GOLDMINE";AT 4,3;"STEHT VOR I
HNEN.";AT 6,3;"WAS WOLLEN SIE MA
CHEN ?";AT 8,3;"1... ZWEI TRAEGE
R ALS KUND-";AT 9,8;"SCHAFTER HI
EINSCHICKEN";AT 11,3;"2... SELBS
T HINEINGEHEN"
5520 IF INKEY$="" THEN GOTO 5520
5530 IF INKEY$="1" THEN GOTO 560
0
5540 IF INKEY$="2" THEN GOTO 570
0
5550 GOTO 5500
5600 CLS
5610 PRINT AT 3,3;"SIE WARTEN CA
. 3 STUNDEN";AT 5,3;"AM MINENEIN
GANG UND SIND";AT 7,3;"DANN SICH
ER,DASS DIE BEIDEN";AT 9,3;"ABGE
STUERZT SIND."
5620 LET T=T-2
5630 FOR N=1 TO 200
5635 NEXT N
5640 GOTO 5500
5700 CLS
5710 PRINT AT 3,3;"SIE GEHEN SEL
BST HINNEIN";AT 5,3;"UND FINDEN
EINEN CLAIM,DER";AT 7,3;"NOCH NI
CHT ABGESCHUERFT IST."
5720 LET S=S+1
5730 FOR N=1 TO 150
5740 NEXT N
5750 GOTO 9
6000 CLS
6010 PRINT AT 2,5;"▒▒▒▒▒";AT 3
,6;"▒█ █";AT 4,6;"▒█ █";AT 5,6
;"▒▒▒▒▒";AT 6,0;"
6015 PRINT AT 8,3;"SIE ENTDECKEN
 EIN DORF";AT 10,3;"UND KOENNEN
HIER EINEN IHRER";AT 12,3;"SCHAE
TZE GEGEN PROVIANT";AT 14,3;"EIN
TAUSCHEN. WOLLEN SIE ?";AT 16,6;
"J=JA/N=NEIN"
6020 IF INKEY$="" THEN GOTO 6020
6025 IF INKEY$<>"J" THEN GOTO 60
50
6030 IF S=0 THEN PRINT AT 18,3;"
SIE HABEN JA GAR KEINEN";AT 20,6
;"SCHATZ..."
6035 IF S=0 THEN GOTO 9
6040 LET P=P+100
6045 LET S=S-1
6050 GOTO 9
6500 CLS
6510 PRINT AT 5,3;"EIN TRAEGER T
RITT IN EINE";AT 7,3;"TIGERFALLE
 UND STIRBT..."
6520 LET T=T-1
6530 FOR N=1 TO 100
6540 NEXT N
6550 GOTO 9
7000 CLS
7010 PRINT AT 3,3;"IN EINER LAGU
NE AM STRAND";AT 5,3;"FINDEN SIE
 KOSTBARE";AT 7,3;"KORALLEN..."
7020 LET S=S+1
7030 FOR N=1 TO 100
7040 NEXT N
7050 GOTO 9
7500 CLS
7510 PRINT AT 3,6;"█████";AT
4,6;"█████";AT 5,6;"█████"
;AT 6,6;"█████";AT 7,4;"
7520 PRINT AT 9,3;"SIE FINDEN EI
NEN PIRATEN-";AT 11,3;"SCHATZ...
";AT 13,3;"ABER DURCH DIE ANSTRE
NGUNG";AT 15,3;"BEIM GRABEN WERD
EN EINE MENGE";AT 17,3;"VORRAETE
 VERBRAUCHT..."
7530 LET S=S+1
7540 LET P=P-50
7550 FOR N=1 TO 200
7560 NEXT N
7570 GOTO 9
8000 CLS
8010 PRINT AT 6,3;"SIE WERDEN VO
N EINGEBOHRENEN";AT 8,3;"UEBERFA
LLEN..."
8020 FOR N=0 TO 40
8030 NEXT N
8050 PRINT AT 12,3;"IHRE SCHAETZ
E WERDEN GERAUBT";AT 14,3;"UND D
IE HAELFTE IHRER";AT 16,3;"MAENN
ER GETOETET..."
8060 LET S=0
8070 LET T=INT (T/2)
8080 FOR N=1 TO 150
8090 NEXT N
8100 GOTO 9
8500 CLS
8510 PRINT AT 3,3;"AUF EINER ANH
OEHE ENTDECKEN";AT 5,3;"SIE EINE
N ALTEN INKA-TEMPEL";AT 7,3;"UND
 IM HAUPTGEBAEUDE FINDEN";AT 9,3
;"SIE ZWEI TONSTATUEN,DIE MIT";A
T 11,3;"EDELSTEINEN VERZIERT SIN
D.";AT 15,3;"JEDOCH WIRD EIN TRA
EGER VON";AT 17,3;"EINEN GIFTIGE
N PFEIL";AT 19,3;"GETROFFEN."
8520 LET S=S+2
8530 LET T=T-1
8540 FOR N=1 TO 200
8550 NEXT N
8560 GOTO 9
9000 CLS
9010 PRINT AT 3,3;"DAS GRABMAL E
INES HOHEN";AT 5,3;"PRIESTERS LI
EGT VOR IHNEN.";AT 9,3;"SIE OEFF
NEN ES UND FINDEN";AT 11,3;"JUWE
ELE UND SMARAGTE,DIE";AT 13,3;"A
LS GRABBEIGABE MIT BEGRABEN";AT
15,3;"WURDEN..."
9020 FOR N=0 TO 100
9030 NEXT N
9040 PRINT AT 17,3;"JEDOCH LAG E
IN FLUCH AUF DEM";AT 19,3;"GRAB
UND DIE HAELFTE IHRER";AT 21,3;"
TRAEGER STARB AN COLERA..."
9050 LET S=S+2
9060 LET T=INT (T/2)
9070 FOR N=1 TO 100
9080 NEXT N
9090 GOTO 9
9990 CLS
9991 PRINT "CASSETTENREKORDER EI
NSCHALTEN...DANN....N E W L I N
E..."
9992 IF INKEY$="" THEN GOTO 9992
9993 SAVE "EXPEDITION"
9995 RUN
```

Unser Weihnachts-Paketservice

Für unsere Leser, die Überraschungen lieben, hat sich unsere Redaktion etwas einfallen lassen. Überzeugen Sie sich selbst. Wir haben Ihnen einige Super-Angebote zusammengestellt:



Spitzen-Programme zum Spitzen-Preis:

Bestell-Nr. 100 **kleines Programm-Paket** 3 bespielte Kassetten DM 24,50
3 bespielte Disketten DM 40,--

Bestell-Nr. 110 **großes Programm-Paket** 8 bespielte Kassetten DM 49,50
8 bespielte Disketten DM 99,--

Bestell-Nr. 200 **exklusive Disketten-Box** inkl. 8 bespielten Disketten zum Preis von DM 130,–

Bestell-Nr. 210 **exklusive Disketten-Box** inkl. 10 Leerdisketten zum Preis von DM 99,–

Alle bespielten Kassetten und Disketten wurden unserem Kassettenservice Seite 84 und 85 entnommen. Unser Diskettenangebot gilt nur für Commodore 64 und Atari!

Bestell-Nr. 300 **Computronic-Angebot** unsere Ausgaben Heft 4–10 zum Sparpreis von DM 25,–

Bestell-Nr. 350 **Abonnement „Computronic"** zum Preis von DM 55,–

Bestellungen, die bis zum 21. 12. 1984 bei uns eingehen, werden rechtzeitig zum Weihnachtsfest geliefert.

Computronic Bestellkarte-Geschenkaktion '84

Die Zustellung erfolgt: gegen **Vorkasse** ☐
(Ausland nur gegen Vorkasse)
innerhalb von 1 Woche

oder Inland per **Nachnahme** ☐
+ Versandkosten

Entnehmen Sie bitte aus unserer Preisliste die notwendigen Angaben für Ihre Bestellung:

Bitte liefern Sie mir:

____
Bestell-Nr. Anzahl

____
Bestell-Nr. Anzahl

zum Preis von gesamt ______________ DM

Name/Vorname: Straße, Nr.:

PLZ/Ort: Datum, Unterschrift

Computer-Börse

Verkäufe

Atari 800XL. Software Liste anfordern gegen 80-Pf.-Briefmarke. Textverarbeitungsprogramm. Hans-Jürgen Brand, bei Wulff, Keselstr. 21, 3000 Hannover 91

Anti-Brumm-Heiz-Adapter im formschönen Slim-Line-Gehäuse für den geplagten Spectrumfreak: 20 DM (Scheck o. Schein). Info gegen 80 Pf Rückporto. T. Schröer, Hohlstr. 11, 6791 Steinbach

Verkaufe Reset-Taste für C 64 und VC 20, ohne Einbau, nur einstecken. 1 Stück 10,– DM, 3 Stück 20,– DM (Schein), Turbotape für C 64 auf Kassette für 30,– DM, 4 eigene Spiele K. 10,– DM. Best. bei S. Valentino, Frankfurter Str. 285, 63 Gießen

TI 99/4A + Exbasic + div. Bücher + Rec. + Kabel + Basic-Kurs, VB 750,- DM.
Tel.: 0 28 61 / 48 45
Sa. ab 18.00 Uhr

Atari/Spectrum-User-Club, monatl. Info, viele Superprogramme, Beitrag nur 5,– DM. Melden bei: ASUC, Gütersloher Str. 68, 4803 Steinhagen

Apple II: E. T. (Alleslerner) o. Geldspieler. Tamm, Solmsstr. 43, 1 Berlin 61

Apple II Super-Disketten-K + ility m. fantastischen Möglichkeiten. Info g. Freiumschlag bei: L. Düning, Kurze Brehe 10, 4500 Osnabrück

Verk. ZX 81 (Fol.-Tast. Def.) + 16K + Netzt. + Koaxkabel + Rec.-Kab. + deut. Anltg. + Buch M.80 Prgr., zus. 120,– DM. Chr. Lenk, Langlütjensand 4, 28 HB 66, Tel. 04 21 / 58 73 26

Verk. **Colour-Genie 32K,** mit original Software NW 600,–, 3 Handbüchern u. Datenrecorder für nur 550,–. Tel. 0 47 21 / 4 72 80

Verkaufe Super-Grafik-Abenteuer „Jetset-Freddy". Mit Highscoreliste, 2 Spielstufen und 6 Höhlen (auf Cassette)! 15,– DM an Bernd Schissler, Hölderlinstr. 3, 7117 Bitzfeld

Verk. TI 59 + Drucker PC100C + Anleitungen + Netzteil + 60 Magnetk. + 2 Rollen Papier, zus. für 450,– DM. W. Lenk, Langlütjensand 4, 28 HB 66, Tel. 04 21 / 58 73 26

Akustikkoppler C-64 DM 138,–, einfach an Userport stecken! Kein Interface nötig. Inkl. Software! Telefonwahlautomat DM 39,–. Info von I. Drust, Landwehrstr. 5, 6100 Darmstadt

Verkaufe: TI-99/4A + Ext. Basic + Joysticks + Recorder + Kabel + TI-Invader + dt. Handbuch + ca. 70 PGM. + 1 TI-Magazin + Chip-Buch. VB: 700,– DM. Tel.: 02 21 / 60 76 76

Achtung: TI99/4A + Mini Memory, Editor/Assembler/Disassembler. Paket nur 30,– DM in Scheinen an T. Mielke, Falkenkamp 17b, 2000 Norderstedt 1

Amateurfunk- und andere Programme für **Spectrum** u. **ZX 81.** Ausführliche Listen gegen Rückporto. Bitte Computertyp angeben! Michael Schramm, Freiligrathstr. 5, 2300 Kiel 1

Verk. TI-99/4A + Ex-Basic + Sprach-Synthesizer + Joyst.-Adapter + ca. 100 Prg. + Zubehör. VB. A. Neuburger, Feldstr. 16, 7505 Ettl. 8, Tel. 0 72 43 / 2 91 75.

TI 99/4 A Ext.-Spiel nur für Erwachsene – bitte Alter ang. zum absoluten Preishit von nur DM 10,- im Umschl. an T. Karbach, Remscheider Str. 18, 5650 Solingen 1

Wenn ihr „Mugsy von Melbourne House" wollt, dann kriegt ihr es für 35 DM (vielleicht) noch bei: Jan Weigner, Carl-Schurz-Str. 11, 2800 Bremen 1, Tel. 04 21 / 34 63 87

TI 99/4A. Erstelle Druckerlisting v. Ihren Programmen in TI- u. ext. Basic, je Pgm.-Zeile 4 Pf. + 2,50 DM f. Portokass. m. V.-Scheck o. Briefmarken an H. Schwab, Loch 21, 8504 Stein

TRS-80/VG Programmliste gegen Freiumschlag. Scheve, 6842 Riedrode

D-64 Forth. *** 49,– DM, Handb. 25,– DM *** Info: D. Luda, Staudinger Straße 65, 8 München 83 ***

★★★ FLOPPY 1541 6 x SCHNELLER ! ★★★ durch Parallelbus für C 64. Interface kpl. geprüft mit Software nur DM 179,–. Info gegen DM 1,50 in Briefmark., B. Akesson, Pf. 802, 4040 Neuss.

Ankäufe

TI-99/4A
Suche **Extended Basic Modul** evtl. Tausch gegen **Editorassembler** sowie Maschinensprachprogramme z. B. **Flugsimulator** und **Ex. Basic Spiel Miner 2049** (USA-Version).
Tel. 0 71 32 / 3 76 08

Suche
VC-20-Programme
Fritz, Wichdorfer Str.14
3501 Emstal 2

Apple II. Suche Software, evtl. auch Tausch. Schickt Eure Liste an Jörn Lubkoll, Boothstr. 23 b, 1000 Berlin 45 *** nur Disk ***

Suche: Gute und billige Software auf Diskette für 64K Atari 600XL. Angebote an: Michael Ebeling, Riepener Straße 7, 3061 Beckedorf

Suche **Atari Diskettenstation,** Aussehen egal, Hauptsache sie funktioniert. Bitte anrufen unter Berlin 0 30 / 4 93 – 19 27

TI 99/4A Minimemory + Logo gesucht. Tel. 08 81 / 6 21 64

Apple II. Bin an Soft- und Hardware für Apple II aller Art interessiert. Ulrich Elsen, Am Jahnplatz 6/383, 4500 Osnabrück

Kontakte

Hallo Spektrumfreaks. Ich möchte mit euch gerne Prog. tauschen. Liste, ob groß oder klein, an: Robert Weidinger, Schönmetzlerstr. 10, 8050 Freising oder Sams.– Diens. 0 81 61 / 43 72

Kontakt mit MZ700-Leuten gesucht. Tel. 05 11 / 42 29 75

***** Atari-Spectrum-User-Club *****
monatl. Info, bundesweit organisiert, riesige Prog.-Bibliothek usw., melden bei:
ASUC, Am Schürhof 17
4905 Spenge
Tel. 0 52 25 / 20 68

Comp.-Club
Bad Hersfeld
sucht Mitglieder!
Kontakt über
Roland Reyer
Am Giegenberg 21
6438 Ludwigsau 1
oder Jugendhaus Hersfeld

Tausch

TI-99/4A Super Ext. Basic und Assembler-progr. Tausch bzw. Verkauf Info (1,–): Stefan Schäfer, Max-Planck-Str. 8, 4050 M.-Gladbach 2. PS: Tauschliste beilegen (nur Assembl.).

Tausche/Verkaufe Atari Prg. 16 K. Uwe Schöneberger, Sonnenstraße 18, 6652 Bexbach 2

Suche Software aller Art (Disk.), für Atari 800 XL u. Antic-Hefte bis März 1984, evtl. auch Computer u. Analog. Dietmar Levenhagen, Landwehr 83, 3057 Neustadt 1, Tel. 0 50 32 / 6 11 81

ZX-Spectrum
Software und Interessenaustausch
Monika Baumgartner
4021 Linz
Postfach 142 Austria
Bitte Liste beilegen

Computer-Börse

Verschiedenes

ZX 81: Schluß mit den Clive-Sinclair-Gedächtnisminuten!! **DD** lädt Ihre Programme **12x** schneller!! Für 20,– DM von: DD-Software, Bernadottestr. 45, 6 Ffm. 50, ☎ 0 69 / 58 71 79

UFO-Forschung. Informationen über UFOs etc. kostenlos. Suche Interessenten zum Aufbau einer Computer-AG (c 64), spez. für wiss. UFO-Forschung! H. W. Peiniger, PF 2361, 5880 Lüdenscheid

VZ 200 Softw. Info g. Freiumschlag. W. Fischer, Friedrichstr. 14, 6463 Freigericht 5

„Load ZX81" lädt jedes ZX81-Prog. mit Variablen **in jeden SPECTRUM.** Die meisten Basic-Programme sind sofort lauffähig. Info anfordern oder **sofort bestellen: DM 39,–.** Michael Naujoks, Rottmannstr. 40, 6900 Heidelberg, Tel. 0 62 21 / 4 68 85

TI 99/4 A, Software, Anwenderprogramme, Spiele und individuelle Programme. Info gegen Rückporto bei: JHC-Software, Chattenstr. 28, 6331 Katzenfurt

C-64-Synthesizer: ca. 40 K, echte Noten, Partiturediting, Notenausdruck, Direkteinspielung + Notenumsetzung, Supersound. Info von Autor W. Kracht, Espellohweg 38, 2000 Hamburg 52

Roulettesoftware f. Sharp PC 1211/12/4551/PC 1500. Roulettesystemprogramme »Auf Dauer gew.« »AD Astra« »Number one« «Acclaim« »Plein Prive« »Accord« DM 550,- Cass. B. Nowotny, Giselastr. 3, 8000 München 40

An alle SF- und Fantasy-Freunde! Jeweils 8 Mitspieler für Strategiespiele gesucht. Mit Gewinnchancen. Näheres gegen Freiumschlag von: H. Topf, A.-Bucherer-Straße 63, 5300 Bonn 1

ZX-SPECTRUM! Spielend BASIC lernen Ca. mit 10 Pr.+Listing+Erklärung+Tips+Tricks für den ZX. 10,- DM bei A. Meuser, Goethestr. 18, 5205 St. Augustin NEU - SUPER ZX Spiele -

Spectrum Softw. z. B. M. Miner 8,-; Z00M 8,-; Liste + kostl. Programm anfordern, Karte genügt: H. Novak, Egerländer Pl. 19, 6053 Obertshausen 2

TI-99/4A: ca. 90 TI-Basic und ca. 80 Ex-Basic Programme zu verk. Info g. Rückumschlag. B. Knedel, Tulpengasse 16, 3171 Weyhausen, Tel. 0 53 62 / 7 11 87

Kleinanzeigen

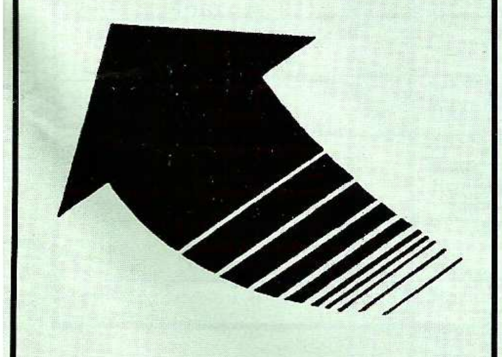
Liebe Leser,
wollen sie ein Kleinanzeige aufgeben? Suchen Sie einen gebrauchten Home-Computer, Software oder Kontakte zu Computer-Clubs etc.?

Dann nutzen Sie jetzt den Kleinanzeigenservice von „Computronic". „Computronic" ist die Zeitschrift für Home-Computer-Besitzer. Sie erreichen durch uns eine Vielzahl von Lesern im In- und Ausland.

Und so wird's gemacht:
Kleinanzeigen-Coupon ausfüllen (alle Zeilen können ausgefüllt werden) und gegen
Vorkasse einsenden an:

Tronic-Verlag
Postfach 41
3444 Wehretal 1

Also, schnell ausfüllen und einwerfen in den nächsten Postkasten. Dann ist mit Sicherheit Ihre Kleinanzeige dabei.



Achtung!

Korrektur Heft 10:

ZX-81 Panik-Labyrinth

Geben Sie bitte in Zeile 1 neun beliebige Zeichen ein, z. B. 1 Rem xxxxxxxxx. Vor dem Starten und Abspeichern folgende Pokes eingeben: Poke 16514,42, Poke 16515,14, Poke 16516,64, Poke 16517,126, Poke 16518,79, Poke 16519,6, Poke 16520,0, Poke 16521,201

Computronic Kleinanzeigen nur gegen Vorkasse

Bitte veröffentlichen Sie ab nächstmöglicher Ausgabe unter der Rubrik:

☐ Verkäufe ☐ Kontakte ☐ Geschäftsempfehlungen
☐ Ankäufe ☐ Verschiedenes ☐ Tausch

folgenden Text:

Bitte jeweils 28 Buchstaben pro Zeile! Bitte Absender nicht vergessen!

Preis für Kleinanzeigen: Private Anzeige DM 4,50 je Zeile
gewerbl. Anzeige DM 8,00 je Zeile
Chiffregebühr DM 10,00 je Anzeige

Name/Vorname:

Straße, Nr.: PLZ/Ort:

Datum, Unterschrift

privat ☐ gewerbl. ☐
(bitte ankreuzen)

Kassettenservice

HEFT 4

VC-64 K = 8,--DM
D = 15,--DM
Mauern, Widerstand

ZX-Spectrum K = 12,--DM
Inventur

TI 99 K = 8,50DM
Drei-Kronen-Spiel
Zahlenputzen

VC-20 K = 11,50DM
D = 18,--DM
Hangman, Saurer Regen, Quadr. Gleichung

Dragon 32K = 8,--DM
Blizzard

Apple II K = 14,50DM
D = 19,50DM
Wilder Westen, Karambolage, Maskengenerator

Atari K = 10,50DM
Mastermind, Schlangenkrieg

HEFT 5

TI 99 K = 14,50DM
Karl der Käfer
Alien-Landing

VC-64 K = 15,50DM
D = 23,50DM
Space-Comets/Erdspalte/Sprite-Data

Apple II K = 9,50DM
D = 19,50DM
Musik-Maker/Mission-Adler/Disk-Catalog

Sharp MZ 700 K = 8,50DM
Kalender
Sharp PC 1500 Lotto

Dragon 32 K = 10,--DM
Space-Flight, Geosoft

ZX-81 K = 10,--DM
Go-Ball, Grand-Prix

ZX Spectrum K = 8,50DM
Missile-Comment

Atari K = 12,50DM
Tank-Battle/Oil Panic

HEFT 6

VC-64 K = 16,50DM
D = 23,50DM
Autostart/Bestellschein/Roadpainter

Dragon 32/64 K = 8,50DM
Wargames

Apple II K = 12,50DM
D = 19,50DM
Snake/Super Datei/Shapetables

VC-20 K = 8,50DM
D = 15,00DM
Bestellschein/Glücky

ZX-81 K = 10,--DM
Moon-Crash/ZX-Draw

ZX-Spectrum K = 13,50DM
Defender/Lui der Wurm/Alternativer Zeichensatz

TI-99 K = 14,50DM
Jack the Digger/Noah - 2099

HEFT 7

VC-64 K = 15,50DM
D = 19,50DM
Hardcopy/Space-Fighter/Data-Generator

ZX-81 K = 10,--DM
Tonprogramm/Aldebaran

Atari K = 12,50DM
Startup/Zeilen-Split/Tomstone-City

VC-20 K = 11,--DM
D = 15,50DM
Multigraph/All-Rammer

ZX-Spectrum K = 12,50DM
Matheprogramm/Bongo-Beecatcher

Apple II K = 12,50DM
D = 19,50DM
Library/Fight

Dragon 32K = 8,50DM
Laser-Attack

TI-99 K = 14,--DM
Lift Bär D = 19,50 DM
ASC II DEF Teil 1

Computronic Bestellkarte-Kassettenservice

Alle im Heft abgedruckten Programme können als zusätzlicher Service über den Verlag bezogen werden.
(Ausland nur gegen Vorkasse)

Die Zustellung erfolgt: gegen **Vorkasse** ☐ innerhalb von 1 Woche

oder Inland per **Nachnahme** ☐ + Versandkosten

Entnehmen Sie bitte aus unserer Preisliste die notwendigen Angaben für Ihre Bestellung:

Bitte liefern Sie mir:

☐ Kassette für ______ (System, aus Heft) ☐ Anzahl

☐ Diskette für ______ (System, aus Heft) ☐ Anzahl

zum Preis von gesamt ______ DM

Name/Vorname: ______ Straße, Nr.: ______

PLZ/Ort: ______ Datum, Unterschrift ______

Kassettenservice

Jedes Programm in Computronic wird einer Endkontrolle in unserem Hause unterzogen und genauestens geprüft. Alle im Heft abgedruckten Programme sind nach der richtigen Eingabe der Listings auch funktionsfähig. Viele Leser verlieren jedoch schnell die Geduld am Programmieren, sollte etwas einmal nicht klappen. Die häufigste Ursache von Störungen im Programm, sind unterlaufene Fehler bei der Eingabe. Verzweifeln Sie nicht, sollten Sie einmal keine Zeit zum Programmieren haben oder sollte etwas nicht gelingen. Alle Programme werden im Verlag gespeichert und können jederzeit mit beiliegender Bestellkarte bezogen werden.

Tragen Sie bitte alle notwendigen Angaben in die Bestellkarte ein.

Rückgabe-Garantie:

Wir garantieren:

- kostenlosen Umtausch von defekten bzw. transportgeschädigten Datenträgern!
- die Zusendung der Umtauschware erfolgt **noch am Posteingangstag!**

Bestellung per Telefon:

Wenn es schnell gehen soll ... rufen Sie uns an. Wir nehmen Ihre Bestellung gern entgegen.

Tel.-Nr.: 0 56 51 – 4 06 93 oder
0 56 51 – 4 06 43
Tronic-Verlag, Postfach 41, 3444 Wehretal 1
nach 17 Uhr:
Anrufbeantworter 0 56 51 – 4 06 93

Bitte beachten Sie:

Sie ersparen sich zusätzliche Kosten (bis zu DM 5,–), wenn Sie per Vorkasse (bar, Verrechnungsscheck) bestellen.

Ausland:

- Bestellung nur gegen Vorkasse!

Disketten für TI-99
* laufen nur mit Speichererweiterung!

HEFT 8

C-64	K =	16,50 DM
	D =	23,50 DM
Monster Attack/Block-Painter/Epson-Drucker		
Atari	K =	14,-- DM
	D =	19,50 DM
Painter/Hardcopy		
Apple II	K =	14,-- DM
	D =	19,50 DM
Reversal Disk-Menue-Generator		
TI-99	K =	14,50 DM
	*D =	19,50 DM
Maya/ASC II DEF Teil 2		
ASC II DEF Teil 1+2	D =	19,50 DM
Laser 2001	K =	8,50 DM
Andromeda		
ZX-81	K =	10,-- DM
Irrgarten 3 D		
ZX-Spectrum	K =	14,50 DM
Solitaire/Superstat. Kleinstes gem. Vielfaches		
VC-20	K =	11,-- DM
	D =	19,50 DM
Zyklo/Meteorit		
Dragon	K =	13,-- DM
Hardcopy		

HEFT 9

Colour Genie	K =	10,-- DM
Fuchs und Hund		
C-64	K =	16,-- DM
	D =	23,50 DM
Projekt Datenbank		
Atari	K =	14,-- DM
	D =	19,50 DM
The Big Quest Fünf gewinnt		
Apple	D =	19,50 DM
Diamonds Hilfsprogramm		
TI 99	K =	14,50 DM
	*D =	19,50 DM
Transfer Silverspar		
Laser 2001	K =	12,-- DM
Crazy Cake		
ZX-81	K =	10,-- DM
Reversi		
ZX-Spectrum	K =	14,50 DM
Jump about		
VC-20	K =	14,-- DM
	D =	19,50 DM
Garten Schloß Gruselstein		
Dragon	K =	10,-- DM
Anwenderprogramm		

HEFT 10

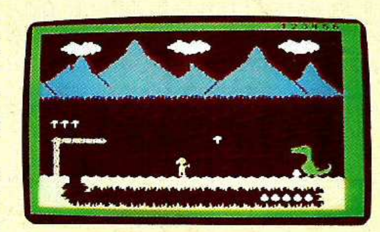
Ti-99	K =	14,50
	D =	19,50
Mother-Duck Screen-Designer		
VC-64	K =	16,50
	D =	23,50
Spiders The Basic		
Atari	K =	11,--
	D =	18,50
Splitt		
ZX-Spectrum	K =	14,50
Pac-Man Oil-Panic		
VC-20	K =	14,--
	D =	19,50
Fressmann Outlaw		
Apple	D =	19,50
Tic – Tac – Toe Jumper		
ZX-81	K =	10,--
Panik Labyrinth		
Dragon 32/64	K =	14,--
Dragon paint		
Laser 2001	K =	8,50
Cave-Man		

HEFT 11/12

VC-64	K =	17,50
	D =	23,50
High Noon Skeet Grafik-Designer		
TI-99	K =	14,50
	*D =	19,50
Cave-Man Alkoholverbot		
Atari	K =	14,50
	D =	19,50
Ski Mutation		
ZX-Spectrum	K =	16,--
Frogger		
ZX-81	K =	10,--
Expedition		
VC-20	K =	14,--
	D =	19,50
Prost Buffalo Bill		
Apple	K =	14,50
	D =	19,50
Donovan Basic-Konverter		

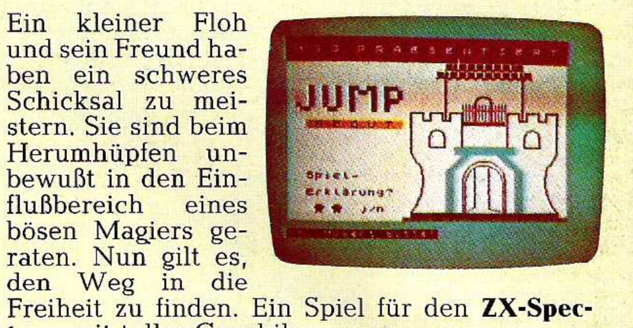
Kassettenservice

Cave Man:



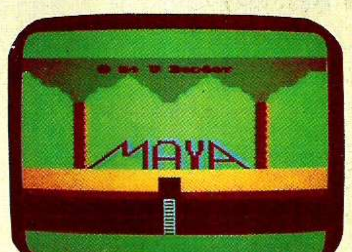
Leo Neandertal lebt in der Steinzeit. Schwer hatten es die Menschen in der Steinzeit, denn sie mußten sich ihr Essen bitter erkämpfen. Leo will dem Saurier Rex die Eier stehlen und diese in den unterirdischen Höhlen lagern. Der Saurier Rex aber hat etwas gegen Leos Unterfangen und versucht mit allen Mitteln, Leo Neandertal am Diebstahl zu hindern. Ein Spiel für den **TI 99.**

Jump about:



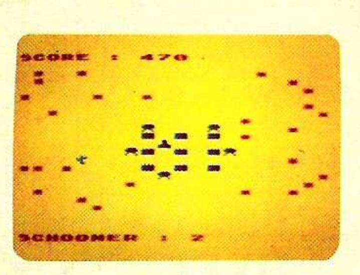
Ein kleiner Floh und sein Freund haben ein schweres Schicksal zu meistern. Sie sind beim Herumhüpfen unbewußt in den Einflußbereich eines bösen Magiers geraten. Nun gilt es, den Weg in die Freiheit zu finden. Ein Spiel für den **ZX-Spectrum** mit toller Graphik.

Maya:



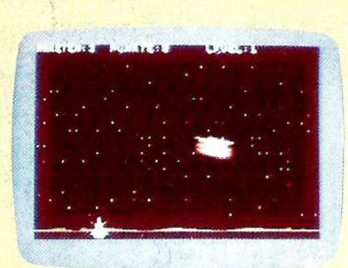
Das Spiel läuft auf dem **TI-99/4 A** mit dem **Extented Basic Modul** und wird mittels Joystick gesteuert. Der Rechner kann in diesem Spiel bis zu 378!! verschiedene Bildschirmszenen darstellen. Das Spiel verfügt über eine hervorragende grafische Darstellung. Begleiten Sie also Coconut Joe auf die abenteuerliche Expedition in den Urwald. Vielleicht haben Sie Glück und finden die Schätze der Mayas.

Tomstone-City:



Eine kleine Stadt im „Wilden Westen" gewährt Dir Schutz vor Deinen Feinden. Solange Du sie nicht verläßt kann Dir nichts passieren. Außerhalb der Stadt wirst Du jedoch gejagt. Gewählt werden kann zwischen verschiedenen Schwierigkeitsgraden. Ein Spiel für **Atari 600/800 XL.**

Monster-Attak:



Für den **Commodore 64.** Fremde Wesen greifen die Erde an. Versuchen Sie die angreifenden Monster vor Erreichen der Erde zu zerstören. Das Spiel unterteilt sich in 6 Level, wobei bei jedem Level die Spielgeschwindigkeit sowie die erreichbare Punktzahl ansteigt. Das Spiel wird mit Joystick gespielt.

In eigener Sache:

Wir haben uns verändert!

Unter dieser Rubrik, liebe Leser, wollen wir Ihnen unser neues Konzept für das kommende Jahr 1985 vorstellen. Nachdem wir nun schon seit April dieses Jahres auf dem Zeitschriftenmarkt erfolgreich mitmischen, können wir eine erste Bilanz ziehen. Das Resultat wird – wie wir meinen – viele Leser mit Sicherheit erfreuen.

Fest steht: Wir bleiben unseren treuen Lesern und Hobby-Programmierern erhalten, aber: in neuer Form!

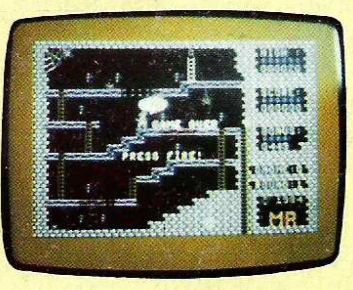
Unser Schwerpunkt wird weiterhin bei Software-Listings der Spitzenklasse liegen. Berücksichtigt werden dabei: *Commodore 64, VC-20, TI-99, Atari, Apple II, ZX-Spectrum* und der *ZX-81.* Zusätzlich erhält der Commodore 64 in jeder unserer Ausgabe einen umfangreichen *Sonderteil!*

Ferner ist Ihnen sicherlich nicht entgangen, daß die jetzige Ausgabe von „Computronic" eine Doppelausgabe ist. Aus Gründen der *zu kurzen Angebotszeit* unserer Zeitschrift im Handel haben wir uns entschlossen, alle 2 Monate auf dem Markt zu erscheinen. Ihr Plus dabei: Wir bringen acht Seiten mehr, Aktuelles, viele Listings, komplette Kurse und andere tolle Überraschungen zum Preis von nur DM 6,50.

Wir jedenfalls hoffen, mit unserem Konzept dem Markt gerecht zu werden, und wünschen uns, liebe Leser, für jeden immer etwas Interessantes dabeizuhaben.
Ihre Redaktion.

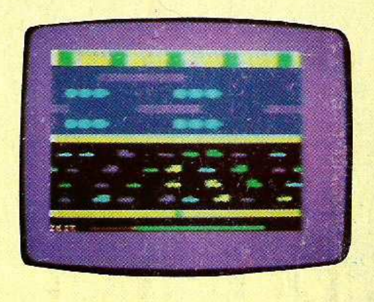
Kassettenservice aus unserem Angebot Seite 84 und 85

Spiders:



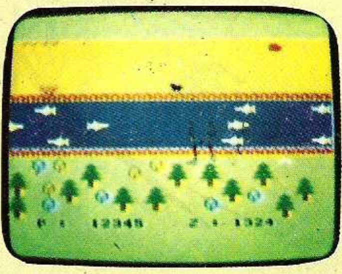
Ziel des Spieles ist es, zwei Spinnen, die auf ihren Netzen herumkrabbeln, zu beseitigen. Vernichten kann man die Tierchen allerdings nur mit Insektengift. Davon liegt noch genug im Keller, aber wo ist der Schlüssel? – Unser Topprogramm aus der Ausgabe Oktober. Spiders gefällt durch eine gute Grafik und guten Sound. Gespielt wird mit Joystick an Port 2. **Für Commodore 64.**

Frogger:



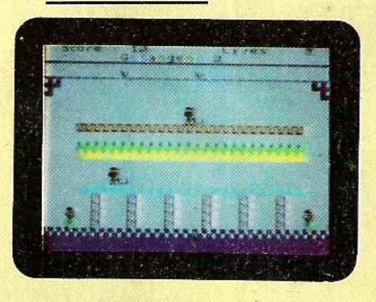
Ein Spiel für den **ZX-Spectrum 48K.** Die beliebte Spielversion jetzt auch für den ZX-Spectrum. Ein Frosch hockt am Straßenrand einer viel befahrenen Straße und versucht verzweifelt, sie zu überqueren. Ist ihm das gelungen, wartet eine zweite schwere Aufgabe auf ihn. Ein Fluß, verseucht mit Krokodilen, muß überquert werden. Das Spiel verfügt über einen guten Sound.

Mother Duck:



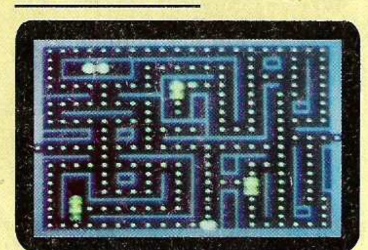
Ähnlich der Spielversion „Frogger". Eine Entenmutter muß einen Fluß überqueren, um Futter für ihre Jungen zu bekommen. Allerlei Untier hält sich jedoch im Wasser auf und macht jede Flußüberquerung zu einer gefahrvollen Angelegenheit. **Für TI 99.** Benötigt werden Joystick und das **TI.-Ext.-Basic-Modul.**

Oil-Panic:



Sie müssen versuchen, möglichst viele Tropfen aufzufangen, die aus einer defekten Ölleitung heraustreten. Aber aufgepaßt, denn der Eimer, der zum Auffangen da ist, ist mit nur vier Tropfen gefüllt. **Für ZX-Spectrum 48K.**

Fressmann:



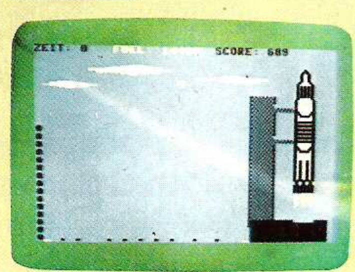
Fressmann ist eine Pacman-Version. Das Programm, welches **ohne Erweiterung** lauffähig ist, steht der Originalversion in nichts nach. Fressmann läuft auf dem **VC 20** und wird mit Joystick gespielt.

Skeet: (Tontaubenschießen)



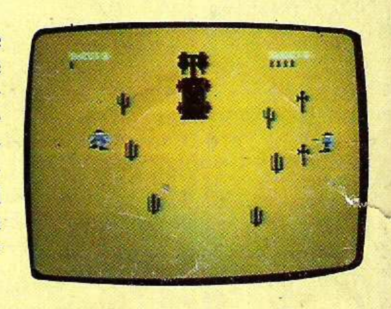
Für **Commodore 64.** Dem realistischen Tontaubenschießen nachempfunden. Von einem Katapult geschleuderte Tontauben müssen reaktionsschnell getroffen werden. In diesem Spiel kann jeder seine eigene Meisterschaft austragen. Eine Supergrafik zeichnet besonders aus.

Projekt:



Als Topprogramm auserwählt von der Redaktion. Sie sollen eine gefährliche Mission ausführen. Um für die nächste Zeit genügend Uran zur Verfügung zu haben, sollen Sie eine Reise zum Mond unternehmen und nach dem edlen Metall suchen. Ihre Reise ist in fünf Phasen aufgegliedert, die nacheinander bewältigt werden müssen. Ein schönes Spiel für den **Commodore 64.**

High Noon:



Ein tolles Spiel für den **Commodore 64.** High Noon besitzt eine schöne Grafik und eine hohe Spielgeschwindigkeit. Gewählt werden kann zwischen 255!! verschiedenen Spielstufen. – Auf einer vielbefahrenen Postkutschenstraße stehen sich zwei Cowboys gegenüber und duellieren sich. Ein Spiel für alle Western-Freunde.

Computronic erscheint jetzt alle 2 Monate im Tronic-Verlag. Der Bezugspreis beträgt im Inland DM 6,50. Im Preis ist die gesetzliche Mehrwertsteuer enthalten.

Tronic-Verlag GmbH · Postfach 41 · 3444 Wehretal 1 · Telefon 0 56 51 - 4 06 43 / 4 06 93